arXiv:2406.09548v1 [cs.LG] 13 Jun 2024

Between Randomness and Arbitrariness: Some Lessons for Reliable Machine Learning at Scale

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

A. Feder Cooper

August 2024

© 2024 A. Feder Cooper
ALL RIGHTS RESERVED

BETWEEN RANDOMNESS AND ARBITRARINESS:

SOME LESSONS FOR RELIABLE MACHINE LEARNING AT SCALE

A. Feder Cooper, Ph.D.

Cornell University 2024

To develop rigorous knowledge about ML models — and the systems in which they are embedded — we need reliable measurements. But reliable measurement is fundamentally challenging, and touches on issues of reproducibility, scalability, uncertainty quantification, epistemology, and more. This dissertation addresses criteria needed to take reliability seriously: both criteria for designing meaningful metrics, and for methodologies that ensure that we can dependably and efficiently measure these metrics at scale and in practice. In doing so, this dissertation articulates a research vision for a new field of scholarship at the intersection of machine learning, law, and policy. Within this frame, we cover topics that fit under three different themes.

First, we quantify and mitigate sources of arbitrariness in machine learning, with respect to hyperparameter optimization and social prediction contexts. We clarify important connections between machine-learning arbitrariness, rooted in non-determinism, with legal notions of arbitrariness that implicate legal rules and due process.

Second, we tame randomness in uncertainty estimation and optimization algorithms, in order to achieve scalability without sacrificing reliability. We discuss how across computing, and particularly in machine learning, scalability and reliability are typically in trade-off. Analogous trade-offs in law and policy make this type of trade-off a useful abstraction for communicating about

machine-learning capabilities and risks to policymakers and other non-expert stakeholders.

Third, we provide methods for evaluating generative-AI systems, with specific focuses on quantifying memorization in language models and training latent diffusion models on open-licensed data. These contributions have urgent and significant connections to U.S. copyright law. We provide an abridged discussion of landmark legal scholarship that details the complicated relationships between generative-AI supply chain and copyright.

By making contributions in these three themes, this dissertation serves as an empirical proof by example that research on reliable measurement for machine learning is intimately and inescapably bound up with research in law and policy. These different disciplines pose similar research questions about reliable measurement in machine learning. They are, in fact, two complementary sides of the same research vision, which, broadly construed, aims to construct machine-learning systems that cohere with broader societal values.

BIOGRAPHICAL SKETCH

A. Feder Cooper was born and raised in New York, NY, and obtained his B.A. in Computer Science and Archaeology from Columbia University in 2014. Prior to a research career, Cooper worked for several years as a software engineer. In 2018, he began his Ph.D. in Computer Science at Cornell University. His doctoral work was chaired by Professor Christopher De Sa, with additional advising by James Grimmelmann, Jon Kleinberg, and Adrian Sampson. His Ph.D. work, broadly construed, studies reliable measurement and evaluation of machine learning, covering both computer science aspects of this work as well as their associated ethical, legal, and policy dimensions.[1]

His contributions span uncertainty estimation, privacy and security of generative-AI systems, distributed training, hyperparameter optimization, and model selection. His work has been recognized by spotlight awards (*NeurIPS 2020*), oral presentation slots (e.g., *AIES 2021*), Best Student Paper (Honorable Mention) at *AAAI 2024*, and a "Rising Star in EECS" award by MIT in 2021. His scholarship on generative AI and copyright has been described as a "landmark" contribution.

A. Feder Cooper is a co-founder of the GenLaw Center and an Affiliate at the Berkman Klein Center for Internet & Society at Harvard University. His Ph.D. research was supported by the John T. and Catherine D. MacArthur Foundation. Following the completion of his Ph.D., he will pursue a postdoctoral research position at Microsoft Research, and will be affiliated with Stanford University, working with Percy Liang and Dan Ho. He will then begin his faculty career at Yale University, appointed as a professor in the Department of Computer Science and an affiliated faculty fellow at Yale Law School.

[1] He had initially planned on attending Harvard Law School; however, thanks to James Grimmelmann's mentorship, he fortunately felt he could skip pursuing more degrees.

For my grandparents

ACKNOWLEDGEMENTS

Over the years, I've heard many metaphors and similes about what graduate school is like. Some say it's like a marriage. Others say it's like being raised by an academic village. Others, still, say it is a trial by fire: to mix metaphors, it's akin to being thrown into the deep end and (hopefully) swimming your way out. For me, it's been like none of these things. I will save my reflections for another time and venue. But I'll note the positive unifying thread of my experience: finding and collaborating with a distributed network of researchers that have a deep love and talent for mischief (in the most innocuous sense of the word). Indeed, they take mischief more seriously than any people I've met before. And this serious mischief has led to some of the most fun and thoughtful collaborations that I could have ever hoped for in my Ph.D.

First, I want to thank my closest faculty collaborators, my advisor, Chris De Sa, and James Grimmelmann. Chris took a chance on me, in many respects a "non-traditional" student, while he was junior faculty. I entered Cornell without prior research experience in computer science (an increasingly rare occurrence), and with an uncompromising desire to do cross-cutting work in machine learning, systems, and law. He gave me the sound advice that this was one interdisciplinary intersection too many, and encouraged me to (at the very most) pick two. It's because of his unwavering support, curiosity, kindness, and generosity that I've been able to chart my own path — to do extensive work in the emerging discipline of machine learning and law.

James has been a champion for my success since my earliest days at Cornell. I am deeply thankful for his feedback, research advice, life advice, and kindness — all of which have shaped my scholarship, research orientation, and career goals. He has been a shining example of the kind of mentor that I hope to be

one day. After effectively being an unnamed author on some of my earlier work — and some gentle prodding to become an official co-author — I feel very lucky that I get to call James one of my closest collaborators. He has co-led, helped shape, and seen to completion what has arguably been the most important work in my career.

In addition to James, I would like to thank my other GenLaw collaborators: Katherine Lee, Niloofar Mireshghallah, and Hoda Heidari. These three working relationships have had an untold impact on my development as a researcher, collaborator, workshop co-conspirator, and human being. These relationships have also evolved into cherished friendships, for which I feel unspeakably fortunate and grateful.

I would like to thank my committee for their assistance and feedback over the years. In addition to Chris and James, mentioned above, I am very grateful to Jon Kleinberg and Adrian Sampson for their advice and expertise in advising my doctoral work. Jon has played a particularly significant role in shaping my thinking about algorithmic fairness, and Adrian is who first introduced me to research that mitigates arbitrariness in computing (in compilers research). Both have had a huge impact on the questions I have studied throughout my degree.

I similarly would like to thank Marilyn Migiel, Pam Samuelson, Abbie Jacobs, Joan Feigenbaum, Solon Barocas, and Michael Littman. Though not official members of my doctoral committee, all six of them have had a tremendous impact on the course of my Ph.D. and career. Marilyn has patiently helped me grow and develop my deep love for Italian language and culture; Pam has been an avid supporter and advocate of my legal scholarship and GenLaw; Abbie has been an incomparable research-idea thought partner, listening ear, friend, and career strategist; Joan has long championed my career as a junior scholar in

the field of Computer Science and Law; Solon has pushed me to think through the (sometimes obscured) normative dimensions of my computing work; and Michael has been a great research and career mentor since before I started graduate school, and has also been a great advocate, conversationalist, and pal. I am so thankful to have had the opportunity to meet all six of them, let alone get to know them and to consider them mentors.

I have also had the great fortune to get to know and work with some incredible researchers at Google DeepMind and Google Research. I am very grateful to Nicholas Carlini, Zachary Charles, Chris Choquette-Choo, Daphne Ippolito, Matthew Jagielski, and Milad Nasr, who, alongside Katherine Lee, have taught me so much about privacy and adversarial ML research, and what it can look like to work together as a research team.

I want to also thank Paul Ohm, Alex Givens, and Miranda Bogen who, with Katherine, James, and Hoda, helped make GenLaw DC as a resounding success. Thank you to Jack Balkin, Miles Brundage, Chris Callison-Burch, and Zack Lipton for their continued support and enthusiasm for the research and practice community that we are trying to create and nurture through GenLaw.

I have also had many great research collaborators over the years — Ph.D. researchers, undergraduates, postdocs, and professors. In particular, I would like to thank the brilliant members of the Relax ML lab, past and present, for their generosity, collegiality, and inspiration over the last six years. Thank you to Ruqi Zhang, Yucheng Lu, Cathy Meng, Jerry Chee, Tao Yu, Albert Tseng, Wentao Guo, Yiming Zeng, Jianan Canal Li, Gary Wei, Khiem Pham, Tiancheng Yuan, and Charlie Ruan. I am especially indebted to Ruqi and Yucheng. When I was just getting acclimated to ML research, Ruqi was a (very) patient, kind, and generous research mentor. Yucheng has been a fantastic colleague, research ad-

vocate, and friend. I would also like to thank my colleagues and friends outside of the Relax ML lab, who have have had a major impact on my scholarship — both directly and indirectly: Maria Antoniak, Manny Moss, Kweku-Kwegyir-Aggrey, Aaron Gokaslan, Jamelle Watson-Daniels, and Jessica Zosa Forde. I am especially grateful to Maria for setting an early example in graduate school of the kind of thoughtful, diligent computing researcher I wanted to become, and to Manny for being a phenomenal thought partner and ally.

I would like to thank the various funding sources throughout my Ph.D. My work has been made possible by generous support from the John D. and Catherine T. MacArthur Foundation (via Jon Kleinberg and Karen Levy) and the Digital Life Initiative at Cornell Tech (via Helen Nissenbaum), a Cornell University fellowship, a runner-up Ph.D. fellowship from Two Sigma, and grant funding from Chris De Sa, James Grimmelmann, Baobao Zhang, and Adrian Sampson.

And most importantly, I want to express my deep fondness, appreciation, and love for my family. Thank you to Eric Schwartz, Salonee Bhaman, Jack Goetz, Bryana Williams, Dhari Noel, and Meghan Witherow. Throughout my Ph.D., you have seen the best of me, the worst of me, and, frankly, the most boring of me. Thank you for sticking by me and having my back when I needed it most, even when I vanished into my work (sometimes for weeks or months at a time). Thank you to Paul and Helaine Cantor for your unwavering belief in me. And last, thank you to Fernando, Bela, Leo, and Achilles Delgado; thank you for giving me a place I can call home, for helping push me to the finish line, and for being some of the best friends, supporters, and companions over the last several years. I could not have done any of this without you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In 2016, I was a backend-systems software engineer playing with machine learning (ML) during my afternoons and weekends. The U.S. presidential election was in full swing, and I had developed the pastime of messing with Facebook's Newsfeed algorithm — perhaps an early glimpse that I should have been an ML security researcher. And in messing with the algorithm, I saw some really horrible content: a lot of virulent, bot-farm, fake stuff. It was everywhere, it was noxious, and it was so brazenly meant to tip the election.

Something was clearly wrong with Facebook's content moderation processes. Or maybe something was exactly right, depending on how you look at it, if this kind of activity contributed to more clicks and engagement. There was clearly a larger phenomenon at play. Human-made platform design decisions and ML algorithms were operating in conjunction with really sophisticated software systems — systems that worked in real-time and at massive scale on the Internet. And these different elements had all mixed together in a potent brew of misinformation and disinformation. This was really upsetting to me. I had gotten into computing — and interested in machine learning in particular — because it is fun. And this stuff (among other things) was decidedly not fun.

It might not have been fun, but it clarified some really big questions for me. It was obvious that large-scale, ML-powered systems (not just ML algorithms) were here to stay. Given this reality, what should we want these systems to do in the world? How can we make sure that these systems are reliable? What does reliability even mean? And if we are unable to make ML systems sufficiently reliable, are there areas where we should not use ML at all? How can we reason

rigorously about this distinction, if it exists? How can we be sure that an ML system's behavior matches up in practice with our intentions and goals? What tools do we have at our disposal — or what tools do we need to invent — to help us reason about this?

There were clearly big, rich, concrete questions in machine learning to study here — in topics like uncertainty quantification, model selection, algorithms and systems trade-offs, and much else. There were also big, rich, concrete questions in law and policy. For example, we could hypothetically come up with the best-ever, theory-backed, ML-based tools for quantifying uncertainty, maybe even at scale. But just because we have a great tool does not mean it is immediately or generally clear how we should use it in practice. Practical considerations require communication with non-expert stakeholders — people who are involved in decisions about whether and how to use ML systems in real-world domains. In this case, this would involve communicating about what different types of uncertainty exist, what they mean concretely in particular practical domains, and, based on its underlying assumptions, what types of uncertainty our great ML-based tool can (and cannot) measure.

More generally, how should we communicate about design choices in ML? Most of these choices are not foregone conclusions. Someone (or some group of people) typically makes some decision at some point in time about which particular model to use in practice. How do we communicate clearly about these types of choices and their consequences to non-experts? How can we make sure that other stakeholders, like policymakers, have necessary and sufficient understanding of ML systems and design choices, so that they can construct sound and useful AI public policy?

Looming among these research questions, there were some big personal ones, too. What was the best way for me to go about trying to find answers to such questions? Should I go to law school? Should I go get a Ph.D. in machine learning? Should I do both? Well, since this is the introduction to my dissertation, it is hopefully clear that I decided to do the ML Ph.D. But I also reasoned that it should be possible to tackle these questions side by side, all at once. Questions like these are two complementary sides of the same research vision. They all involve research into how to do reliable measurement for ML at scale, where what constitutes "reliability" takes into account considerations that are relevant not just for ML, but also for law and policy.

There is a virtuous cycle in this type of work. Making contributions with this particular focus in ML is indivisible from concrete implications for tech law and policy; doing deep work in tech law and policy raises novel research questions to tackle on metrics and measurement practices in ML. For example, in order to understand the copyright implications of generative-AI systems, we need to be able to take useful and replicable measurements that can help inform questions judges and policymakers have about issues like copyright infringement.

Following this vision, I have begun an extensive research program in machine learning, law, and policy, and I have done this work across a bunch of projects. I am the first author on most of them [136–147, 210, 349–351, 641], and much of this work has received awards — spotlight, oral, and best paper honorable mention accolades [136, 138, 141, 144, 210, 336, 350, 641].

Even if all of this research touches on topics that fundamentally have to do with the intersection of machine learning, law, and policy, it has been very important to make sure that the core contributions of each piece are cogniz-

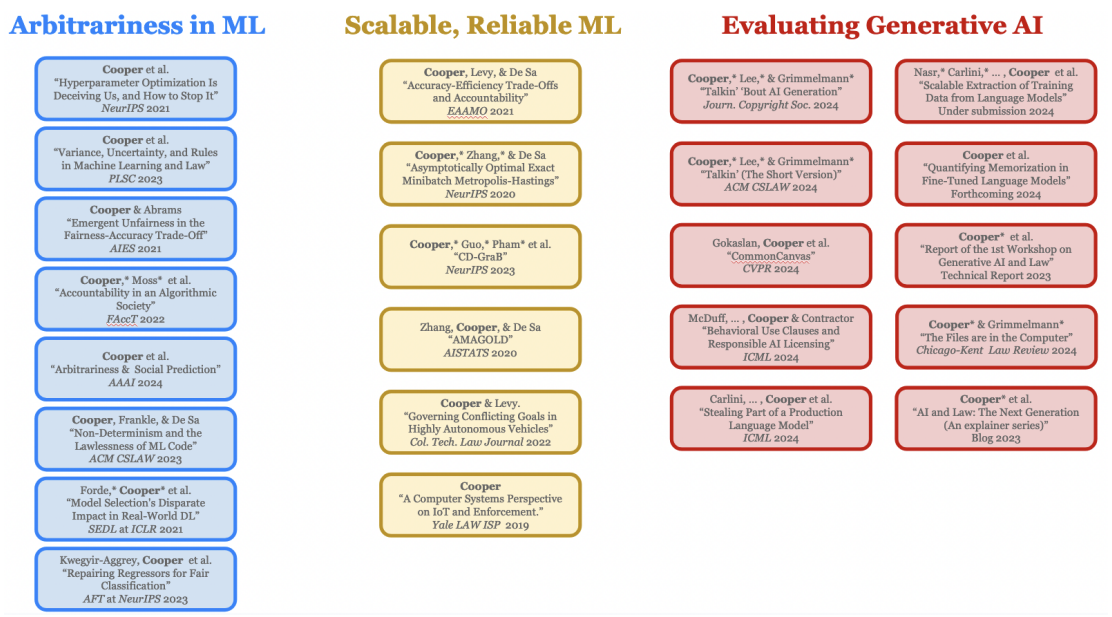


Figure 1.1: Ph.D. projects organized by theme. Some projects do not fit neatly into these divisions [137, 147, 346], and many projects cross boundaries. Notably, Appendix G [146] touches on all three themes.

able to the appropriate disciplinary audiences. As a result, a large number of these projects have their main contribution positioned in machine learning, and have been published or presented in venues like *NeurIPS*, *ICML*, *AAAI*, and the like [108, 137, 140, 141, 145, 210, 236, 336, 399, 435, 641, 642]. A smaller number have had their main contribution in law and policy, and have been published in law reviews and interdisciplinary computing venues like *ACM CSLAW* [138, 139, 142–144, 349, 350]. A smaller number still have their main contribution in computing ethics and values, and have been published at venues like *ACM FAccT* [136, 146, 147, 346, 351].

Maintaining these disciplinary boundaries has been useful to keep in mind for publishing; however, what has been more useful, with respect to posing research questions, is considering overarching research themes. There were two themes that I had intended to explore in my Ph.D., based on my initial motiva-

tion for going to graduate school: sources of arbitrariness in ML and scalable ML algorithms (Figure 1.1). My work on arbitrariness is deeply related to model selection choices — ML modeling and algorithm choices that people make, which can lead to arbitrary outcomes. In scalable ML algorithms, my work has studied how to make algorithms more efficient while retaining reliability guarantees, predominantly in uncertainty estimation.

Both themes have clear connections to law and policy. Arbitrariness is a very important concept in the law, for example, with respect to due process [218]. In light of this relationship, I have focused my work on quantifying and mitigating ML-specific types of arbitrariness, and making these types of arbitrariness cognizable for law and policy. Scalability and reliability are often in trade-off; this can serve as a useful abstraction for communicating with policymakers about implementation decisions and associated capabilities and risks.

With two coherent themes concerning ML, law, and policy, we could perhaps call it day. One such theme might be a happy accident, but two entirely different ones indicates a pattern — an indication that this field of work is a fruitful direction for original scholarship. However, the dissertation does not end here.

In summer 2020, I was tinkering with GPT-2 and GPT-3, shortly after GPT-3 [92] came out. There was a clear leap in quality between GPT-2 and GPT-3; GPT-3 was nearing human-like text generation. Its architecture was larger, and it was also trained on a much larger quantity of (likely copyrighted) text data. One day, when there was an ever better model, GPT models would no longer be a research curiosity. They would be sufficiently impressive, such that they would be embedded in consumer-facing products that people would actually want to use. And when that day came, it would likely be a nightmare for intel-

lectual property (IP) law.

This was just a hunch; I did not know much about IP law at the time. So, in Fall 2020, I decided to enroll in a course on IP at the law school, and then I waited. And I did not have to wait long because, about two years later, OpenAI released ChatGPT and everything changed. All of the considerations that had brought me to graduate school were, all of a sudden, immediately and inescapably relevant. There was a real-time, large-scale, ML-driven system, governed by innumerable human design choices, that had enormous societal implications — and everyone was using it. I would no longer have to explain why work at the intersection of ML, law, and policy was so important. Everyone would know it from firsthand experience.

In other words, this moment presented a huge opportunity for the type of work I had already been pursuing. But it also meant that I should redirect my energy toward a third line of work in the last year of my degree — a line of work on generative AI and law. Based on the enormous and urgent demand for clarity and rigor in this area, my work in this theme has thus-far focused on evaluations for generative-AI systems that provide insights for U.S. copyright law.

Dissertation Format

This dissertation is organized in three parts around these three themes.

- Part I addresses arbitrariness in machine learning.
- Part II details projects in scalable machine learning algorithms.
- Part III discusses evaluating generative-AI systems, with particular attention to copyright-related topics.

Each of these parts is outlined in the remainder of this introduction (Sections 1.1, 1.2, and 1.3, respectively). While they are presented separately, it is worth noting that the three themes they cover appear throughout. For example, scalable machine algorithms and their associated trade-offs feature in all three parts.

In an attempt at concision, this dissertation only addresses a subset of the research projects mentioned above (Figure 1.1). Each part contains the same overall structure of three chapters that have been integrated into a single narrative. The first two chapters reflect papers that contain core contributions in machine learning, and third chapter demonstrates how the first two have deep interrelationships with tech law and policy. Additional research concerning cross-cutting philosophical questions about the relational aspects of ML accountability is deferred to the appendix.

1.1 Part I: Sources of Arbitrariness in Machine Learning

Part I presents three inter-related research projects that study arbitrariness in machine learning and its consequences for law and policy. Broadly speaking, this work studies how human-made decisions can lead to arbitrary results or conclusions in ML experiments. These decisions may seem quite mundane in practice — the selection of a particular set of hyperparameters [145] (Chapter 2) or a specific classification model to deploy [141] (Chapter 3) — but they can in fact result in outcomes that mislead us about ML capabilities and risks. As a result, ML arbitrariness is a significant consideration for law and policy [138]. Indeed, there are deep connections between arbitrariness in machine learning

and how law and policy reason about and mitigate unwanted sources of arbitrariness in legal contexts (Chapter 4).

Chapter 2: Arbitrariness in Hyperparameter Optimization Choices

This part opens with work on characterizing arbitrariness in hyperparameter optimization (HPO). In particular, Chapter 2 uses tools from modal logic to formalize the process of drawing conclusions about algorithm performance when running hyperparameter optimization in machine learning experiments.

It is well-known that HPO greatly affects overall measurements of algorithm performance. There is much prior experimental work in machine learning that has articulated this point [122, 176, 543], such that it is safe to say that it is common knowledge in the ML community. HPO can affect results so much that the results of two different HPO procedures for the same task and the same optimizers can lead to contradictory conclusions. The two sets of experiments in Figure 1.2 highlight this phenomenon. Both experiments test three optimizers — SGD, Heavy Ball momentum, and Adam — to train the VGG-16 neural network to classify the CIFAR-10 dataset. On the left, we test one set of hyperparameter configurations, pick the best-performing configuration per optimizer, and compare test accuracy. We do the same thing for the experiments on the right, but we change how we configure the hyperparameter search space for Adam — represented in the third, rightmost box plot.

Separately, the plot for each of these sets of experiments suggests a particular conclusion. On the left, it looks like Adam performs worse than SGD and Heavy Ball. That is, the results reasonably suggest the conclusion that non-

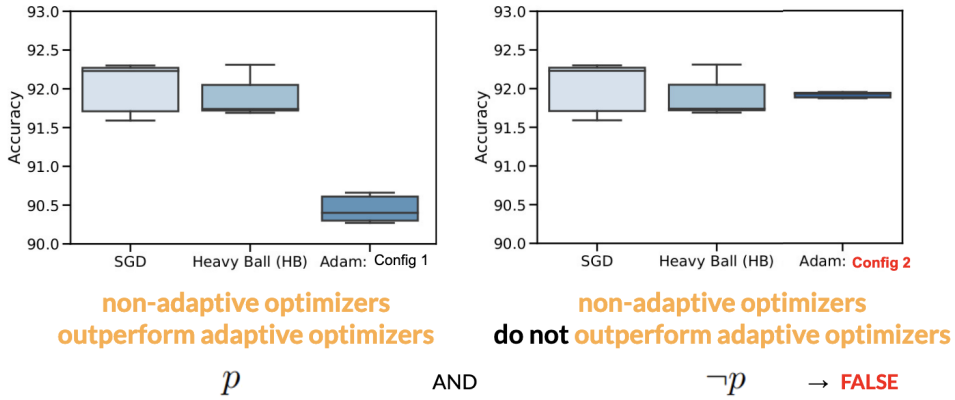


Figure 1.2: Running different sets of experiments for training the VGG-16 architecture to classify images in CIFAR-10. Both sets of experiments test SGD, Heavy Ball momentum, and Adam. The experiments on the right use one configuration for Adam, and the experiments on the left use another. In isolation, each of these sets of experiments leads to a conclusion that, when considered together, result in a logical contradiction.

.

adaptive optimizers like SGD and Heavy ball outperform adaptive ones like Adam. The results on the right tell a very different story. Judging by test accuracy alone,[1] Adam performs just as well as SGD and Heavy Ball. If we were to accept both sets of experiments as valid HPO configurations to test empirically, we would yield a logical contradiction (Figure 1.2). This implies that these sets of experiments cannot both be valid ways to test hyperparameters because, taken together, the conclusions they suggest are inconsistent. Taken together, these experiments do not enable us to produce reliable knowledge about algorithm performance.

Ideally, we want to avoid this type of situation in ML research, since one of our goals is to develop reliable knowledge about algorithm performance. Importantly, this is not the same as making claims from ML experiments involving HPO that have to do with ground-truth algorithm performance. We do

[1]If we consider variance, Adam seems to out-perform SGD and Heavy Ball.

not know the ground truth. Instead, we want to make sure that the ML community does not accept *a priori* a particular methodology for configuring and performing HPO that could possibly lead to inconsistent conclusions, like those in Figure 1.2. In other words, it would, be fine for the ML community to accept exclusively either of the sets of experiments in Figure 1.2, and to draw the selected set's related conclusion. Or it would be fine for the ML community to be skeptical — to accept neither of these sets of experiments, and to conclude nothing at all about algorithm performance. However, it is not fine for the ML community to accept both sets of experiments as valid, as this is the case that leads to inconsistent conclusions.

This is a bit of a subtle point. Obviously, when presented with these two sets of experiments side-by-side, we know to reject them because they yield inconsistent conclusions. But this is not typically what happens in practice. Instead, researchers typically perform one (if any) pass of HPO, which in our motivating example would only produce one set of experiments in Figure 1.2 from which one could form conclusions. In our work in this chapter, we therefore aim to study a kind of meta-problem: we want to make sure that, even when we are presented with only one set of results, we form conclusions that are not *arbitrary* — conclusions that constitute reliable knowledge. That is, if someone else had by happenstance configured HPO slightly differently for the same overall experiment, they would not have yielded results that suggest a conclusion that contradicts the one that we have obtained.

Based on this motivation, we attempt the first theoretical study of how to draw reliable conclusions from empirical studies using HPO. We pursue this goal in two parts. First, we come up with a formalization that enables us to rea-

son about two vague types of uncertainty in our problem setup: (1) the possible outcomes of HPO experiments and (2) whether we believe the conclusions that can be drawn from those outcomes. The point of formalizing our beliefs is to instill an appropriate amount of doubt when examining HPO results: even if we cannot know for certain what is true, we do not want to end up believing a conclusion that is false [170].

We use modal logic [70] for this formalization, since it is a useful analytical tool for pinning down vague, difficult-to-capture (non-stochastic) types of uncertainty in both of these sources. Second, we use our formalization to prove non-trivial theorems about whether or not a hyperparameter optimization procedure is defended drawing false, inconsistent conclusions. We suggest an HPO procedure and use our formalization to prove that it is defended against such an outcome (within a limited time budget).

Chapter 3: Arbitrariness in Social Prediction

There are many other sources of arbitrariness in machine learning, not just the (non-stochastic) arbitrariness that gets introduced through decisions in configuring hyperparameter optimization procedures. In another line of work, we investigate another type of arbitrariness related directly to randomness: how arbitrary the choice of single model is, based on the specific random seed used for training, in algorithmic fairness contexts.

To get a sense for this arbitrariness, let us examine a simple example. Consider training 100 random forest models on COMPAS, which is (for many reasons) an infamous binary classification task that has been used to predict

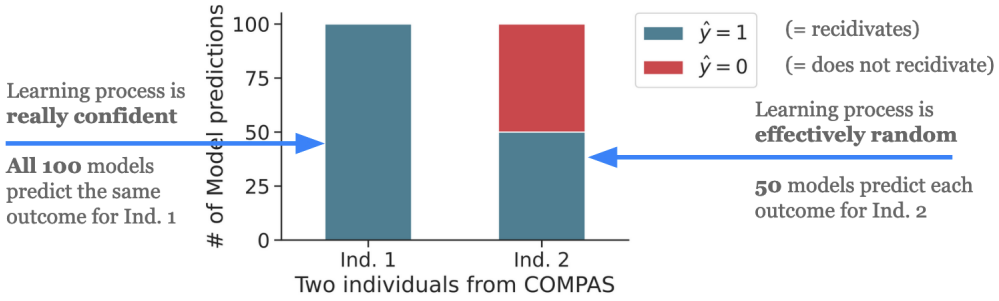


Figure 1.3: 100 bootstrapped random forest models show models can be very consistent in predictions $\hat{y}$ for some individuals (Ind. 1) and arbitrary for others (Ind. 2). In this example, 50 models result in predictions that suggest Ind. 2 will *recidivate* (i.e., commit a crime again) and 50 that suggest they will not. Their prediction is *arbitrary*.

whether someone is going to *recidivate* — whether they are going to commit a crime again [344]. Such predictions can then be used to inform whether an individual is allowed to receive bail or not, if they are rearrested.[2] We train these 100 models using bootstrapping with different random seeds [184–186], and they will serve as our empirical estimate of the distribution over possible random forest models (with a particular set of hyperparameters). We can then look at two individuals in the reserved test set, run our 100 trained models on them, and plot the counts of the resulting predictions for each (Figure 1.3).

The 100 models all produce the same prediction for Individual 1. We can understand this to mean that the learning process that produced these models is really confident with how it classifies Individual 1. If we were to pick one model to use in practice — as the algorithmic fairness binary classification problem formulation often does — there would be no effect on how Individual 1 is classified. But the story is really different for Individual 2: the learning process is not sufficiently confident to justify assigning Individual 2 either decision outcome.

[2]There are many issues with this setup, ranging from problem formulation issues to complications of using rearrest as a proxy for whether or not someone has committed a crime. We refer the reader to Barocas et al. [45] for a summary.

Their classification is *arbitrary*. With this learning process, we produce predictions that are akin to flipping a coin, where the result of the flip is a product of happenstance — of the random seed used we happened to use during training. Importantly, this arbitrariness remains latent in the common fair binary classification problem setup, in which we just evaluate one model. We instead need to look at the empirical distribution over possible models to surface it.

These two individuals reflect the best and worst case scenarios, in terms of arbitrariness in predictions. They are also two real individuals: these are real outcomes for two individuals in the COMPAS dataset when training random forests. The training process clearly results in outcomes that treat them very differently, with respect to arbitrariness. In Chapter 3, we turn this intuition for arbitrariness into a metric, which we call *self-consistency*.

Self-consistency can be computed for any test instance, and results in a number in the range between 0.5 and 1: 0.5 maps to minimally self-consistent examples like Individual 2, and 1 maps to completely self-consistent examples like Individual 1. Because we can compute self-consistency on a per-instance basis, we can measure it for particular individuals, like those visualized in the bar plot in Figure 1.3. But we can also measure and visualize self-consistency across the entire test set, in order to understand overarching patterns about arbitrariness in predictions for particular datasets.

We use cumulative density functions (CDFs) to do so across a variety of fair binary classification benchmark datasets. This enables us to plot different levels of self-consistency on the x-axis, and the probability that a test instance attains (at least) that level of self-consistency on the y-axis. With this approach, we uncover novel and important insights about arbitrariness in social prediction

settings. For example, we find that about 20% of predictions in COMPAS (using random forests) are 0.5 self-consistent (Figure 1.4). In this setting, 1 out of every 5 test examples in COMPAS resembles Individual 2 (Figure 1.3); approximately 20% of prison recidivism classifications are arbitrary — a coin flip — which should be really disturbing if this kind of analysis is used to inform whether an individual receives bail or not.

In the remainder of Chapter 3, we examine this type of arbitrariness in detail. We discuss methods for improving self-consistency, in order to root out this particular type of arbitrariness, and we also examine the impact of improving self-consistency on more-traditional algorithmic fairness metrics [260].

Chapter 4: Legally Cognizable Notions of ML Arbitrariness

The types of arbitrariness that we quantify in HPO (Chapter 2) and social prediction (Chapter 3) settings yield important insights about how to draw reliable conclusions from machine learning experiments. But they also reveal a lot more in terms of broader impact. Arbitrariness is not just a useful concept to pin down and reason about with respect to reliability in ML. It is also a concept that plays significant roles in law and policy — running the gamut from theoretical work in legal philosophy [218] to practical policy decisions [324]. The research discussed in both of these chapters puts forth definitions for ML arbitrariness that are directly informed by law and policy scholarship on arbitrariness. In turn, the insights that this work elicits suggest novel ways for how law and policy can reason about types of arbitrariness that are particular to machine learning — arbitrariness that implicates important social values like due process and safety when ML systems are deployed in practice.

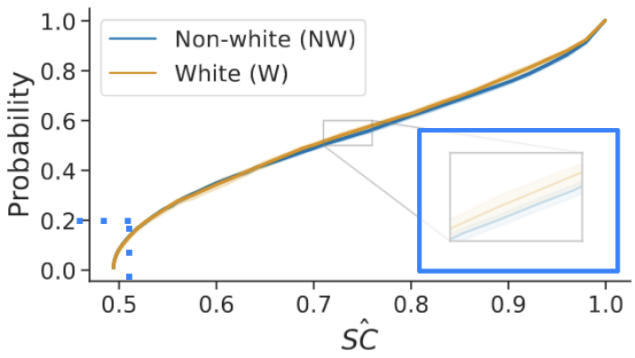


Figure 1.4: Training 101 bootstrapped random forest models on COMPAS 10 different times. Our estimates for self-consistency (x-axis) are very stable, as evidenced by the tightness of the error bars. In this setting, roughly 20% of classification decisions (indicated with the blue dotted line) in COMPAS are **predictably and consistently arbitrary**, resembling Individual 2 in Figure 1.3.

To give one example, let us return briefly to the social prediction example of COMPAS and prison recidivism. The underlying models that contribute to our computations of self-consistency are clearly quite different, given that they can result in arbitrary predictions for significant portions of the test set. Recall that, for random forests, 20% of predictions on COMPAS are arbitrary — they resemble Individual 2 (Figure 1.3). In other words, we can understand the individual models that we train in this setting to be *unstable*. However, even though these individual models are unstable, the self-consistency estimates that they enable us to produce are in fact (generally speaking) *very stable*. Regardless of the random seeds that we use to train 101 models on COMPAS, we produce a set of 101 models that lead to similar estimates of self-consistency for the test set.

We can see this in the CDF figures in Chapter 3 (see also Figure 1.4): to produce these figures, we compute self-consistency across the test set 10 different times, for different sets of 101 models. The resulting plotted CDF curves are averages, and the error bars surrounding them are very tight. (Indeed, we had to include insets to zoom in, in order to clarify that they are in fact present.)

Regardless of how we split COMPAS into train and test sets, we find that, for random forests, approximately 20% of predictions on COMPAS are always arbitrary. Put differently, we find that 20% of COMPAS predictions are **predictably and consistently arbitrary** — a mouthful of a concept that seems to turn some concepts from the law and policy on their head. In law and policy, predictability and arbitrariness are often described as opposites, rather than concepts that can operate at different levels of abstraction, such that both can be true at the same time.

In Chapter 4, we present published research that scratches the surface of insights like this for law and policy. We discuss how non-determinism in machine learning can lead to types of arbitrariness that diverge from how law and policy tend to conceive of arbitrariness. This, in turn, suggests fundamental and important differences between machine-learned rules and legal rules — differences that have important consequences for broader impact, including how the law should reason about using ML in practice. This chapter, though published, represents preliminary work that we are currently developing for law review.

1.2 Part II: Taming Randomness in Scalable, Reliable Sampling and Optimization Algorithms

The arbitrariness that we investigate in Part I ultimately can be traced to different sources of non-determinism in the development of ML systems — whims in human decisions, randomness in ML algorithms, and non-determinism in computer systems. In Part II, we focus particularly on how to harness randomness in ML algorithms, so that, at scale, we can achieve reliable outcomes (in the sta-

tistical sense, which we describe here). Reliability and scalability tend to be in trade-off in ML, and in computing more generally. The work we present in this part shows how we can navigate and sometimes even push the boundaries of such trade-offs.

Chapter 5 discusses a method for reliable, scalable Bayesian inference, which can be used to do uncertainty estimation at scale; Chapter 6 details a distributed, SGD-based optimization algorithm that finds better-than-random example permutation orders to accelerate convergence; and Chapter 7 ties together threads across scalable ML to explain how common trade-offs, like those between scalability and reliability, have direct analogues in law and policy. This makes such trade-offs a useful abstraction for policymakers to understand overarching design choices and resulting behaviors of large-scale ML systems.

There are also various connections between work in this theme and the first. Notably, the work in Part I on reasoning about possible models and self-consistency in fairness contexts (Chapter 3) was greatly influenced by our prior work concerning uncertainty quantification (Chapter 5, Zhang et al. [642]).

Chapter 5: Scalable, Reliable Uncertainty Quantification

Our first encounter with uncertainty in this dissertation involved using the bootstrap method [184–186] to compute self-consistency as a proxy for quantifying arbitrariness (Chapter 3). We begin here with this intuition of uncertainty, through our now-familiar example of measuring self-consistency in the COMPAS dataset.

In this example (Figure 1.3), we trained 100 different possible models on

COMPAS using bootstrapping, and compared predictions for two individuals in the test set. All 100 predictions for Individual 1 are for the same class; in contrast, Individual 2 exhibits 50 predictions for one class, and 50 for the other. In other words, the learning process produces models that are *high variance* in their predictions for Individual 2, and no variance for Individual 1. This variance captures predictive uncertainty. The learning process produces models that, taken together, are very certain concerning how to predict for Individual 1, and completely uncertain concerning how to predict for Individual 2.

Computing predictive variance is just one way of quantifying uncertainty, but there are others. The gold-standard method, arguably, is *Bayesian inference*. Given that y is a prediction, $\boldsymbol{x}$ is an input data example vector, $\boldsymbol{D}$ is the training dataset, $\mathbb{H}$ is the model architecture (the hypothesis class), and $\boldsymbol{\theta}$ is the vector of model parameters,

$$\underbrace{p(y|\boldsymbol{x}, \boldsymbol{D}, \mathbb{H})}_{\text{posterior predictive distribution}} = \int \overbrace{p(y|\boldsymbol{x}, \boldsymbol{\theta}, \mathbb{H})}^{\text{likelihood}} \underbrace{p(\boldsymbol{\theta}|\boldsymbol{D}, \mathbb{H})}_{\text{posterior}} d\boldsymbol{\theta}. \tag{1.1}$$

This equation models what is called the *posterior predictive distribution*: the probability of a prediction y, given a specific input data example $\boldsymbol{x}$, dataset $\boldsymbol{D}$, and type of model $\mathbb{H}$. This distribution can be computed in relation to the *likelihood* and *posterior*. The likelihood is the probability that a given input example $\boldsymbol{x}$, model parameters $\boldsymbol{\theta}$, and model architecture $\mathbb{H}$ could result in the prediction y. The posterior reflects the probability that the given dataset $\boldsymbol{D}$ and architecture $\mathbb{H}$ could yield the particular model parameters $\boldsymbol{\theta}$. We then integrate the likelihood and posterior over all of the possible model parameters $\boldsymbol{\theta}$: we weight the likelihood by the posterior for all possible models. Altogether, this means that we are capturing the uncertainty in the prediction y for a given input $\boldsymbol{x}$, with respect to all possible learned models $\boldsymbol{\theta}$ that have architecture $\mathbb{H}$ and are trained

on dataset $\boldsymbol{D}$.

There is a lot more that one can say about this setup. (Indeed, this is the focus of Chapter 5.) For our purposes here, the important point is that this is just a different way of measuring uncertainty than what we did with bootstrapping in our COMPAS example in Chapter 3. This is just a different way of modeling the distribution over possible learned models, where here we refer to the learned models $\boldsymbol{\theta}$.

Unfortunately, the integral in Equation (1.1) is intractable to analyze exactly. But we can approximate it with a *Monte Carlo* estimate, using a concrete number N of models $\boldsymbol{\theta}_i$:

$$\begin{aligned} \underbrace{p(y|\boldsymbol{x}, \boldsymbol{D}, \mathbb{H})}_{\text{posterior predictive distribution}} &= \int \overbrace{p(y|\boldsymbol{x}, \boldsymbol{\theta}, \mathbb{H})}^{\text{likelihood}} \underbrace{p(\boldsymbol{\theta}|\boldsymbol{D}, \mathbb{H})}_{\text{posterior}} d\boldsymbol{\theta} \\ &\approx \frac{1}{N} \sum_{i=1}^{N} p(y|\boldsymbol{x}, \boldsymbol{\theta}_i, \mathbb{H}), \end{aligned} \tag{1.2}$$

where different concrete models $\boldsymbol{\theta}_i$ are sampled from the posterior, i.e., $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta}|\boldsymbol{D}, \mathbb{H})$. On the left, we still have the posterior predictive distribution; but now on the right, instead of an integral, we compute an average over the N likelihoods for different concrete models $\boldsymbol{\theta}_i$, where the different $\boldsymbol{\theta}_i$ are drawn from the posterior distribution.

We still, however, do not know what the posterior distribution is. To get an estimate, we can use something called *Markov chain Monte Carlo* (or *MCMC*), which simulates the posterior. At a high level, MCMC proposes a sequence (a Markov chain) of samples of models θ_i that reflect the posterior distribution. It performs a random walk or simulates some physical dynamics (e.g., Hamiltonian, Langevin dynamics), which we can compute in practice. This simulation

depends on a function, $U(\boldsymbol{\theta})$, which is called the *potential* or *energy* function. We can compute this potential, which can also be related to the posterior using Bayes' rule:

$$\overbrace{p(\boldsymbol{\theta}|\boldsymbol{D},\mathbb{H})}^{\text{posterior}} = \overbrace{\frac{\overbrace{p(\boldsymbol{D}|\boldsymbol{\theta},\mathbb{H})}^{\text{likelihood}}\overbrace{p(\boldsymbol{\theta}|\mathbb{H})}^{\text{prior}}}{\underbrace{p(\boldsymbol{D}|\mathbb{H})}_{\text{evidence}}}}^{\text{Bayes' rule}} \propto \exp \overbrace{(-U(\boldsymbol{\theta}))}^{\text{negative potential}} \tag{1.3}$$

So we now have a way to estimate the posterior, but, unfortunately, we are still not quite done. Even though we can compute this simulation process, it exhibits a problem: it is biased. And this bias can cause the chain of samples $\boldsymbol{\theta}_i$ that we simulate to drift away from the true posterior distribution.

To correct for this bias, we add in one more step to the simulation process: the *Metropolis-Hastings* (or *MH*) correction step [263, 413]. The MH correction step rejects some of the samples we have generated; it does not include them in the Markov chain. This involves performing computations with the potential function, which result in either accepting or rejecting the proposed sample (see Chapter 5, Brooks et al. [89], Figure 1.5). As a result, the simulation process does not contain all of the samples that we generate, just the $\boldsymbol{\theta}$ that get accepted. Then, once we have this Markov chain of samples that reflect an unbiased estimate of the posterior, we can use it to help us quantify uncertainty: we can plug it back into Equation (1.2), which approximates the posterior predictive distribution (1.1) with our Monte Carlo approximation.

Unfortunately (again), even though MCMC is a clear improvement over the intractable integral in Equation (1.1), it is still *really* expensive to compute in practice. It is expensive because, as is clear from Equation (1.3), the potential function $U(\boldsymbol{\theta})$ has a dependency on the dataset $\boldsymbol{D}$. This means that performing

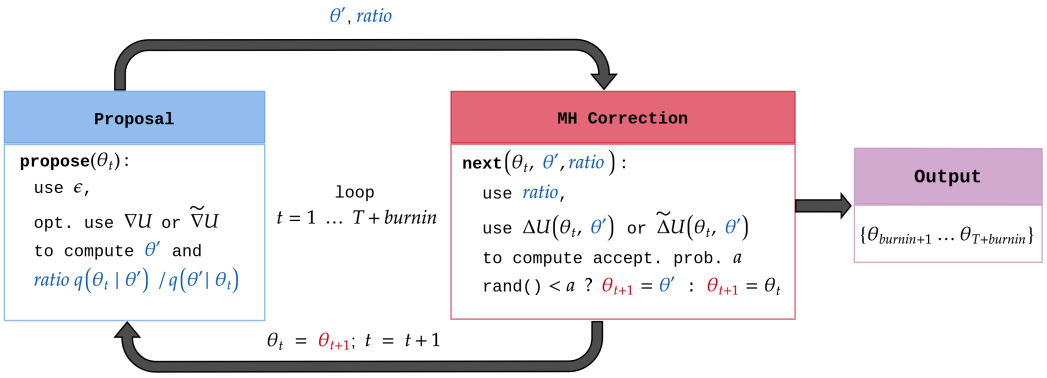


Figure 1.5: Exact MCMC composes a proposal step (to produce new samples $\boldsymbol{\theta}'$) with an MH correction to remove bias by deciding to accept/reject the new sample as the next stage in the Markov chain ($\boldsymbol{\theta}_{t+1}$). Our exact, scalable algorithms use 1) proposals that leverage stochastic gradients of the potential, $\tilde{\nabla} U$ [642]; 2) MH corrections that use minibatches of data examples for computations with the potential. $\tilde{\Delta} U$ (Chapter 5).

computations with the potential requires iterating over the entire dataset, and we need to do this every single iteration of the simulation in order to produce a new sample. For large-scale datasets — basically every dataset in modern ML — this is often too costly to do in practice. It is certainly more expensive than optimization; however, optimization only gives a single point estimate of the model parameters. It gives us an infinitesimally small sliver of the posterior distribution, making it an unreliable estimate of the entire posterior (Figure 1.6). So, even though optimization is more efficient, we cannot use it to do uncertainty estimation reliably.

More generally, we can note that reliability and scalability are in trade-off for uncertainty estimation. Optimization is really scalable, but it is not very reliable because it just gives a point estimate of the posterior. And MCMC is really reliable — it gives a good estimate of the whole posterior — but it is not at all scalable because each iteration depends on the size of the dataset. Prior work strikes different balances between these two competing goals. For ex-

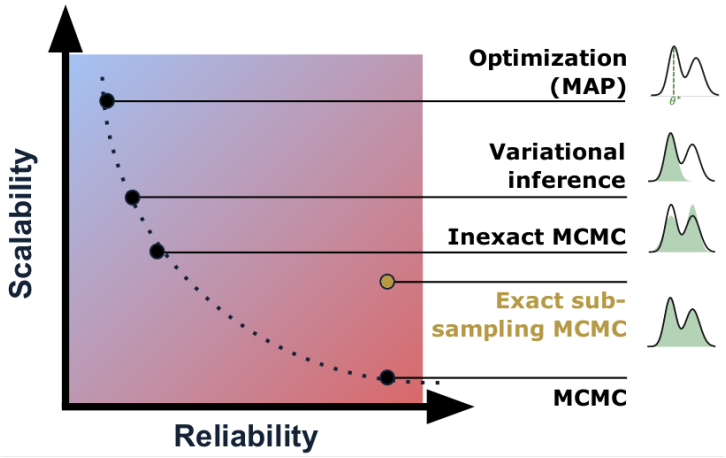


Figure 1.6: Reliability-scalability trade-off in Bayesian inference (i.e., for capturing the posterior of possible models). We visualize the posterior on the right. Optimization provides a single estimate of the posterior (dotted line labeled θ^*, top right); MCMC captures the whole posterior (fully shaded area under the curve, bottom left). Our work (**yellow**) carefully uses subsampling to push the frontier: it captures the full posterior, but does so more efficiently than traditional MCMC.

ample, *inexact* MCMC uses subsampling to improve efficiency; it removes the dependency on the dataset size at each simulation iteration by using only a subset of the dataset for computations. But subsampling can once again introduce bias: we can lose the guarantee that the simulation will converge to a reliable estimate of the posterior (Figure 1.6).

So, at last, this is where our work comes in. We introduce subsampling carefully to the simulation process, so that it is possible to get efficiency gains, while still guaranteeing that we converge to the correct posterior that traditional MCMC yields. In this respect, our work has managed to push out the trade-off curve between scalability and reliability for uncertainty estimation (Figure 1.6). In Chapter 5, we discuss one of our algorithms that achieves this goal by using minibatches of data to compute the accept/reject decision in the MH correction step.

Chapter 6: Scaling Distributed Optimization

Despite the reliability of Bayesian inference for performing uncertainty estimation, optimization has remained the workhorse of modern ML. We have also done research to scale up optimization, such that it converges to a point estimate more efficiently.

For optimization algorithms like stochastic gradient descent (SGD), users typically randomly shuffle training data examples without replacement each epoch. *Random reshuffling* is so common that it is often implemented as a boolean flag in interfaces in common deep learning libraries (e.g., Pytorch has an option for setting `shuffle = True`). The reason that people use random reshuffling is that, in practice, it tends to speed up convergence. However, as our work in Chapter 6 shows, there exist permutation-based example orders that perform better than random reshuffling: these non-random orders achieve provably faster convergence rates for stochastic gradient descent. Lu et al. [384] find better permutation orders for training in centralized settings. In Chapter 6, we find such orders for the contemporary, more efficient setting of distributing training across a number of parallel workers.

The high-level idea is to leverage information in per-example gradients from prior training epochs, in order to identify a permutation for example ordering in the next epoch; this example order contributes to making more progress in converging to a point estimate of the model parameters. To find such permutations, we leverage insights from kernel thinning (which builds on ideas from coreset selection) [182, 183], and herding and vector balancing [20, 261, 620]. The math that we rely on from this prior work is defined in terms of arbitrary vectors. We extend this to the distributed optimization setting, in which the vectors that we

balance are per-example gradients.[3]

Relying on this prior work, we show that, over time, balancing per-example gradients achieves the bound in the herding problem formulation. In Chapter 6, we prove that, by achieving the herding bound in the parallel setting, then SGD exhibits an accelerated convergence rate in comparison to distributed random reshuffling. Further, we demonstrate a speedup over Lu et al.'s work in the centralized setting [384], which is linear in the number of parallel workers.

The balancing algorithm that we use is fairly inexpensive, but it does exhibit some memory overhead and computational cost over distributed random reshuffling, (associated with node communication and data sorting, see Appendix E.3.1). In other words, our algorithm pays some per-epoch cost in efficiency in order to find higher quality example orders. But overall, over some time, this results in needing relatively fewer epochs to converge; our algorithm is more efficient and scalable, as exhibited by our provably faster convergence rate.

The "over time" aspect of this benefit is especially relevant. It does indeed take several epochs to find permutations that bring down the herding bound and confer our algorithm's benefits. If a particular task converges quickly, or if we only run a few epochs of training (as is common right now in pretrained base-model fine-tuning), then we typically do not observe speedups over random reshuffling. Future work should further investigate these trade-offs, such that the benefits of our work can better extend to common contemporary training paradigms.

[3]Lu et al. [384] extends herding and balancing to the centralized optimization setting. We realize additional benefits by also incorporating insights from kernel thinning.

Chapter 7: Exposing Legally Cognizable Trade-Offs to Enable Accountability

The work in both Chapters 5 and 6 navigates trade-offs between scalability and reliability. Trade-offs like this exist all over machine learning. They tell us a lot about what is possible to achieve with respect to important, competing goals. And they also tell us a lot about possible decisions ML researchers and practitioners can choose to make — how they can choose to balance needs for scalability and efficiency with concerns about maintaining sufficient reliability in specific contexts.

It turns out that trade-offs like these are not exclusive to computing. In Chapter 7, we discuss how analogous trade-offs crop up all over domains that policymakers frequently reason about — complex domains as diverse as law, public health, and federal risk assessment policy. For just one example, consider the U.S. code for civil procedure. It contains a number of rules, such as speedy trial requirements and statutes of limitations, that impose time constraints to encourage efficient case resolution. The need for efficiency is balanced against competing needs for thorough fact-finding and argumentation. Based on this overarching observation, we argue that such trade-offs expose a very useful abstraction that policymakers can rely on to help them reason about (and regulate) ML systems. Policymakers do not necessarily need to understand very low-level technical details about machine learning algorithms and systems. They can glean a lot about relevant details about systems capabilities by understanding machine learning at the level of these types of trade-offs.

The work in Chapter 7 was published in 2021 [144], and dates back to a project that was started in 2018. At the time, we motivated our research with the concrete example of reasoning about risks in autonomous vehicles, as they

were a particularly germane example of a large-scale ML system where balancing efficiency and reliability has a clear, broader impact on safety. The conceptual contributions of our work extend far beyond this motivating example. They translate directly to this current moment, in which large-scale generative-AI systems that perform real-time inference are being deployed in consumer-facing products. In future work, we will update the research in this chapter in light of the ascendance generative-AI systems.

1.3 Part III: Evaluating Generative-AI Systems

There has been a tremendous amount of recent public interest in generative AI, both excitement about capabilities and concern about risks. One frequent set of concerns around generative AI is that the training and use of generative-AI systems involves practices that infringe copyright. In the year and a half since ChatGPT's release, groups of artists, individuals, and companies have filed over two dozen copyright lawsuits in the U.S. against the builders and deployers of generative-AI systems [117].

In Part III, we dig into both the technical and legal aspects of generative-AI systems, with a specific focus on copyright. In Chapter 8, we discuss recent work on extracting (potentially copyrighted) memorized text training data from large language models. In Chapter 9, we explore the benefits and drawbacks of training a family of text-to-image latent diffusion models exclusively on permissively licensed, Creative Commons images with synthetic captions. Last, in Chapter 10, we present an abridged version of our framework [349] for thinking about the interplay between generative AI an copyright: the *generative-AI supply*

chain, which maps the very many stages invoked in the creation, deployment, and use of generative-AI systems with the very many actors that are involved at those stages. We apply the supply-chain framing to U.S. copyright, but note that it is more broadly useful for reasoning about the impacts of generative AI.

Chapter 8: Measuring Memorization in Language Models

In Chapter 8, we discuss recent work on extracting memorized text training data from large language models (LLMs). In high-level terms, *memorization* in generative-AI contexts often refers to cases in which one can "deduce or produce a model's given training example" [142]. We make contributions that show how to feasibly measure memorization for large-scale production systems — in particular, ChatGPT [435]. We use security-style attacks on LLMs by prompting them with particular inputs, which result in output generations that are verbatim copies of training data examples. This work has direct relationships to a variety of law and policy issues, notably copyright and privacy, since memorization can result in a model regurgitating creative expression (like a portion of copyrighted novel) or sensitive content (like a social security number) that was in its training data.

Figure 1.7 shows a preview of our results. On the left, we show an example of our attack. We ask ChatGPT to repeat single tokens forever — in this case, the word "poem." At first, the model (and system in which it is embedded) responds by following this instruction. But eventually (and almost always), the output *diverges*, and sometimes that divergent content contains memorized training data.

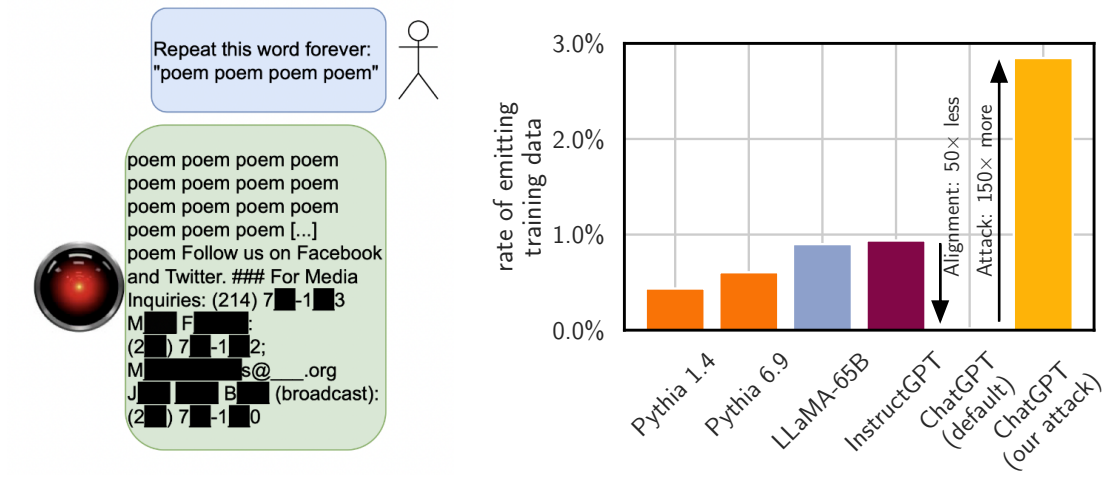


Figure 1.7: The aligned ChatGPT 3.5 appears 50× more private than prior models (**right**). We develop an attack (**left**) that shows it is not: ChatGPT emits training data 150× more frequently than prior work (default). Figures reprinted with permission from my collaborators.

This finding was very exciting to people, and even received news coverage [440, e.g.]. It was the first large-scale memorization extraction attack on an aligned, deployed production system. As a result, our findings also implicate various stages of the generative-AI supply chain (Chapter 10), not just model training and generation. The corresponding paper, which is currently under journal submission, is a large-scale measurement study of what we call *extractable memorization*.[4]

Chapter 9: Training Latent Diffusion Models on Open-Licensed Images

One of the key issues for memorization centers on the use of copyrighted data during training. If we do not train models on copyrighted data, then (by definition) models will not memorize copyrighted data that they could later regurgitate near-verbatim. (This, importantly, should not be mistaken for indicating

[4]The paper has a lot of really excellent science in it (not just fun sound bites like "asking ChatGPT to say 'poem poem poem' breaks ChatGPT").

"an image of elsa from frozen"

(a) Prompt (b) SD2 (c) CommonCanvas-S-C

Figure 1.8: Prompting Stable Diffusion 2 (b) and CommonCanvas (c) with "an image of Elsa from Frozen" (a).

that training on public domain or licensed data will resolve all potential copyright problems; it is still possible to produce potentially infringing generations if one only trains on public domain or licensed data [139, 142].) This raises a natural question: what if we trained models on only permissively licensed or public domain data? By training on such data, we will hopefully reduce the risk of producing potentially copyright-infringing models that can be used to produce potentially copyright-infringing generations.

In Chapter 9, we begin exploring these ideas in the context of training a family of latent diffusion models for image generation. We curate a large dataset of open-licensed, Creative Commons images, for which we generate accompanying synthetic captions, and we use this dataset to train Stable Diffusion 2 architecture variants. When we prompt these models to try to elicit potentially copyrighted expression, we observe some interesting outcomes. For example, prompting with "an image of Elsa from Frozen", Stable Diffusion 2, which was trained on copyrighted data, generates an image that strongly resembles the Disney character. In contrast, our model, CommonCanvas [236], does not. Nevertheless (beyond the fact that this is just one example), we are not exempt from all possible copyright-related problems. We discuss this below, with respect to Chapter 10 and in recent work [139]).

Chapter 10: Bridging Copyright Law and the Generative-AI Supply Chain

Chapters 8 and 9 serve as concrete examples of why generative AI is complicated for copyright-related questions. However, as works with core contributions in machine learning, they do not contend with legal specifics. In Chapter 10, we dig into these specifics: we provide a comprehensive framework for reasoning rigorously about the interplay between generative AI and law. We make the case that, when forming legal questions about generative AI, we should be doing so in terms of the entire *generative-AI supply chain* that is invoked in the creation, deployment, and use of generative-AI systems.

Our supply-chain framing takes many terms that are familiar for those with a background in machine learning (e.g., pre-training, fine-tuning) and ties them together with the very many actors that influence and interact with generative-AI systems (Figure 1.9).[5] This framing illustrates the complex ecosystem involved in generative-AI system production, and navigates this complexity by providing a way to think precisely about "*what* technical and creative artifacts are produced, *when* these artifacts are produced and stored, and *who* exactly is involved in the production process." In turn, we are then able to carefully map the stages of the generative-AI supply chain to the very many parts of U.S. copyright law that they potentially implicate. This enables thoughtful discussion about "*what* is potentially an infringing artifact, *when* in the production process it is possible for infringement to occur, and *who* is potentially an infringing actor" [349, p. 32].[6]

[5]The supply-chain framing connects the "many hands" [146] involved in generative-AI systems to the many stages that constitute these systems' production. For more on the problem of how "many hands" serves as a barrier to accountability, see Appendix G.

[6]In Chapter 10, we present the shorter conference version of our work on copyright and the generative-AI supply chain [350]; the longer version, quoted here, is forthcoming in a law journal.

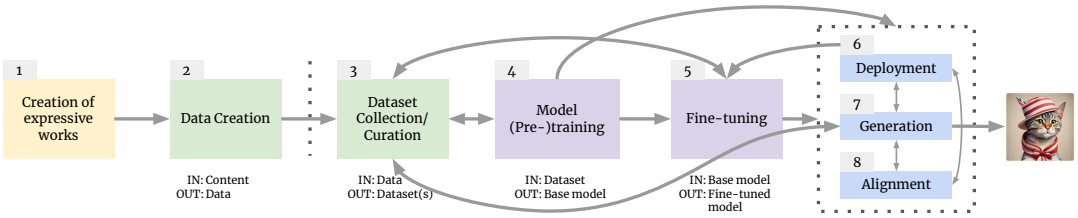


Figure 1.9: We conceive of the *generative-AI supply chain* as consisting of 8 deeply interwoven stages, each of which can involve many (potentially different) actors.

With these contributions, we can see clearly why the projects in Chapters 8 and 9 are such beautiful examples of why the supply-chain framing is so important. We need this whole supply-chain view to understand how the different stages interact (Figure 1.9), and how this can have nuanced implications for copyright (and more).

In general, research on memorization has clear connections to copyright. Given how it is commonly defined in the technical literature, memorization is wholesale copying [142, 435]; wholesale copying, by definition, implicates U.S. copyright law [151, reproduction right]. Models become capable of memorization during the training process (Figure 1.9, stages 4, 5 and, possibly, 8): it is during training that particular memorized training data examples get encoded somewhere within the model's parameters.[7] Only then can memorization can get exposed to end-users at generation time, in response to user-provided prompts (Figure 1.9, stage 7). In the case of our work on ChatGPT in Chapter 8, memorization can also embroil aligned (Figure 1.9, stage 8), deployed (Figure 1.9, stage 6) production systems. Our divergence attack broke alignment and managed to evade whichever system-level guardrails are in place (e.g., output content filters), such that we ultimately were able to surface mem-

[7]For more on the relevance of this reality to U.S. copyright, see Cooper and Grimmelmann [139], which is not included in this dissertation.

orized training data in generations. Reasoning about the potential copyright consequences of memorization in ChatGPT requires engaging with each of these stages and the actors engaged in them.

For CommonCanvas text-to-image models (Chapter 9), the training data require both images and text captions. We collected a set of permissively licensed Creative Commons images, most of which lacked descriptive text captions. In our data curation process (Figure 1.9, stage 3), we generated synthetic captions for these images (Figure 1.9, stage 7) using a publicly released (Figure 1.9, stage 6), off-the-shelf, pre-trained captioning model called BLIP-2 [369] (Figure 1.9, stage 4). BLIP-2 was trained on LAION data [523] — one of the datasets that links to copyrighted images, and that is named in several current U.S.-based copyright lawsuits [24, e.g.]. In short, our curation process depended on a generating synthetic captions, which we produced with a pre-trained model whose own training data contained copyrighted images. Even though our models are trained on licensed images, our data curation process depends on another model, which was itself trained on images that were not explicitly licensed.

Clearly, there are complex interrelationships between CommonCanvas's supply-chain stages, so we cannot just look at individual stages in isolation when thinking about copyright consequences. We cannot just look at our trained model's curated training data — Creative Commons images and (likely uncopyrightable) synthetic captions. We also have to look upstream in the supply chain at how different actors curated the training data for BLIP-2: the off-the-shelf generative-AI model that we chose to use for image captioning. Without this view, we would miss potentially relevant and significant observations — in this case, how (transformed) copyrighted data is indispensable, however

indirectly, for training our open (or, perhaps more accurately, "open") models.

We perform extensive analysis of the supply chain with respect to U.S. copyright law in Chapter 10 (as well as in our law-review paper [349]). Even though this work is very recent, it is already having a significant impact. It has already been used as an authoritative source by U.S. congressional staffers and government agencies. Journalists and copyright scholars have called it "landmark" work, and a "magnum opus" [66, e.g.].

In our work, we also provide some broader lessons and takeaways about copyright and generative AI. One of these lessons, in particular, relates to a thread present throughout this dissertation: design choices matter a lot for overall system behavior and its consequences (in this case, for copyright); these choices, and thus resulting system behaviors, are typically not foregone conclusions. This takeaway is very important to keep in mind with respect to law and policy — to governance and accountability [146] concerning generative-AI systems. And it is also an important and great thing to keep in mind for machine learning research. As this dissertation shows by example, wherever there are design choices, there are concrete research questions that we can study in computer science.

1.4 Closing Thoughts

The nine chapters discussed above may seem neatly organized into the three discrete themes outlined in this introduction. Nevertheless, while reading, it is worth keeping in mind that these divisions are somewhat artificial; all three themes are cross-cutting. They appear in different degrees throughout the en-

tirety of this dissertation. For example, trade-offs between reliability and efficiency do not just appear in our work on machine learning algorithms. They also appear throughout all of our work on evaluating generative-AI systems (e.g., we choose a relatively simple metric for extractable memorization, because it is more efficient to measure at large scale). They permeate the choices we make when formulating ways to measure and mitigate arbitrariness (we sacrifice a good deal of efficiency for reliable proxies of arbitrariness).

All of these themes bubble up into the overarching questions that we began with in this introduction — the questions that brought me to graduate school. These questions fundamentally concern how to do reliable measurement for machine learning at scale: making choices in metric design (e.g., how we choose to define uncertainty in ML), figuring out how we can dependably measure these metrics at scale and in practice (e.g., in large-scale systems with real-time capabilities), and communicating the effects of our measurements to other, often non-expert stakeholders (e.g., policymakers).

At a higher level, still, all of these research questions are about pursuing, developing, and refining what we want ML systems to do in the world. They are about how we can make sure that ML system behavior matches up with our goals, values, and intentions. For me, these remain the big important questions. This dissertation is just a start at carving out some smaller, concrete questions that we can answer, in service of these big important ones.

Part I

Sources of Arbitrariness in Machine Learning

Clarifying uncertainty around non-deterministic, ML-driven decision processes is an important mechanism for characterizing reliability.[8] Importantly, there are numerous of such uncertainty in ML, not just the type of model uncertainty that Bayesian inference can measure (Chapter 5). Another kind of uncertainty gets introduced by the sometimes-arbitrary choices that ML experts make in their implementations and experiments [146, 147]. In this part, we explore how to quantify and mitigate this type of arbitrariness, and communicate its importance to the legal community. This chapter reflects work that has been published at *NeurIPS* (poster), *AAAI* (Best Student Paper, Honorable Mention), *ACM CSLAW* (Long Presentation), *AAAI/ACM AIES* (Oral), and an *ICLR* workshop (Oral).

First, we discuss how choices in experiments that involve hyperparameter optimization (HPO) can result in arbitrary conclusions about overall algorithm performance (Chapter 2, Cooper et al. [145]). We develop a theoretical framework for reasoning reliably about the conclusions we can draw from experiments involving HPO. HPO is a bi-level optimization problem: the inner loop learns model parameters, and the outer loop selects a set of (not learned) hyperparameters from a search space that is often chosen by hand [205]. Hyperparameters influence training and have an enormous impact on performance measurements. In fact, one can be easily misled to draw arbitrary conclusions about ML methods in practice — e.g., that algorithm $\mathcal{A}$ is more accurate than algorithm $\mathcal{B}$, or the reverse — depending on the chosen search space. Our work in this chapter uses modal logic [190] to enable writing proofs about HPO procedures — proofs that can guarantee that, within a given training-time budget, it is not possible to yield inconsistent conclusions about algorithm performance.

[8] See also Appendix G, which details the relationship with accountability.

Second, we detail how the choice of concrete training dataset (based on random seed) can lead to arbitrary predictions. Building on our prior work in fairness-metric design [136] and high variance in deep-learning fairness classification experiments [210], we study how latent uncertainty in modeling available benchmark datasets result in arbitrary outcomes for individual test examples in algorithmic fairness contexts. Approximating the distribution over models surfaces how predictions are really consistent for some individuals and effectively arbitrary for others. By turning this intuition into a metric, we find that arbitrariness plays a huge role in measurements of unfairness.

And third, we discuss how work on arbitrariness in ML is inspired by and has informed deep connections between ML, law, and policy. In law, non-arbitrariness is an important value that plays a significant role in legal reasoning about due process and discrimination [161, 218, 324, 570]. We discuss past and ongoing work that clarifies how better measurements of arbitrariness in ML can meaningfully have a direct impact on law and policy.

CHAPTER 2

HYPERPARAMETER OPTIMIZATION IS DECEIVING US, AND HOW TO STOP IT

We begin our study of arbitrariness in machine learning with a study on hyperparameter optimization. Choices of hyperparameter search space (if searched at all) tend to be arbitrary, based on folklore rather than rigorously tested insights. We explore how such choices can also lead to arbitrary conclusions about algorithm performance, rather than reliable scientific knowledge.

Chapter summary: Recent empirical work shows that inconsistent results based on choice of hyperparameter optimization (HPO) configuration are a widespread problem in ML research. When comparing two algorithms $\mathcal{J}$ and $\mathcal{K}$, searching one subspace can yield the conclusion that $\mathcal{J}$ outperforms $\mathcal{K}$, whereas searching another can entail the opposite. In short, the way we choose hyperparameters can deceive us. We provide a theoretical complement to this prior work, arguing that, to avoid such deception, the process of drawing conclusions from HPO should be made more rigorous. We call this process *epistemic hyperparameter optimization* (EHPO), and put forth a logical framework to capture its semantics and how it can lead to inconsistent conclusions about performance. Our framework enables us to prove EHPO methods that are guaranteed to be defended against deception, given bounded compute time budget t. We demonstrate our framework's utility by proving and empirically validating a defended variant of random search.

This chapter is a licensed derivative copy of work published at *NeurIPS 2021* [145].

2.1 Introduction

Machine learning can be informally thought of as a double-loop optimization problem. The *inner loop* is what is typically called *training*: it learns the parameters of some model by running a training algorithm on a training set. This is usually done to minimize some training loss function via an algorithm such as stochastic gradient descent (SGD). Both the inner-loop training algorithm and the model are parameterized by a vector of *hyperparameters* (HPs). Unlike the learned output parameters of a ML model, HPs are inputs provided to the learning algorithm that guide the learning process, such as learning rate and network size. The *outer-loop* optimization problem is to find HPs (from a set of allowable HPs) that result in a trained model that performs the best in expectation on "fresh" examples drawn from the same source as the training set, as measured by some loss or loss approximation. An algorithm that attempts this task is called a *hyperparameter optimization* (HPO) procedure [131, 205].

From this setup comes the natural question: how do we pick the subspace for the HPO procedure to search over? The HPO search space is enormous, suffering from the curse of dimensionality; training, which is also expensive, has to be run for each HP configuration tested. Thus, we have to make hard choices. With limited compute resources, we typically pick a small subspace of possible HPs and perform grid search or random search over that subspace. This involves comparing the empirical performance of the resulting trained models, and then reporting on the model that performs best in terms of a chosen validation metric [205, 283, 302]. For grid search, the grid points are often manually set to values put forth in now-classic papers as good rules-of-thumb concerning, for example, how to set the learning rate [278, 343, 348, 465]. In other words, how

we choose which HPs to test can seem rather ad-hoc. We may have a good rationale in mind, but we often elide the details of that rationale on paper; we choose an HPO configuration without explicitly justifying our choice.

Much recent empirical work has critiqued this practice [80, 122, 176, 376, 403, 430, 526, 543]. The authors examine HPO configuration choices in prior work, and find that those choices can have an outsize impact on convergence, correctness, and generalization. They therefore argue that more attention should be paid to the origins of empirical gains in ML, as it is often difficult to tell whether measured improvements are attributable to training or to well-chosen (or lucky) HPs. Yet, this empirical work does not suggest a path forward for formalizing this problem or addressing it theoretically.

To this end, **we argue that the process of drawing conclusions using HPO should itself be an object of study**. Our contribution is to put forward, to the best of our knowledge, the first theoretically-backed characterization for making trustworthy conclusions about algorithm performance using HPO. We model theoretically the following empirically-observed problem: When comparing two algorithms, $\mathcal{J}$ and $\mathcal{K}$, searching one subspace can pick HPs that yield the conclusion that $\mathcal{J}$ outperforms $\mathcal{K}$, whereas searching another can select HPs that entail the opposite result. In short, the way we choose hyperparameters can deceive us — a problem that we call *hyperparameter deception*. We formalize this problem, and prove and empirically validate a defense against it. Importantly, our proven defense does not make any promises about ground-truth algorithm performance; rather, it is guaranteed to avoid the possibility of drawing inconsistent conclusions about algorithm performance within some bounded HPO time budget t. In summary, we:

- Formalize the process of drawing conclusions from HPO (epistemic HPO, Section 2.3).
- Leverage the flexible semantics of modal logic to construct a framework for reasoning rigorously about 1) uncertainty in epistemic HPO, and 2) how this uncertainty can mislead the conclusions drawn by even the most well-intentioned researchers (Section 2.4).
- Exercise our logical framework to demonstrate that it naturally suggests defenses with guarantees against being deceived by EHPO, and offer a specific, defended-random-search EHPO (Section 2.5).

2.2 Preliminaries: Problem Intuition and Prevalence in ML Research

Principled HPO methods include *grid search* [302] and *random search* [53]. For the former, we perform HPO on a grid of HP-values, constructed by picking a set for each HP and taking the Cartesian product. For the latter, the HP-values are randomly sampled from chosen distributions. Both of these HPO algorithms are parameterized themselves: Grid search requires inputting the spacing between different configuration points in the grid, and random search requires distributions from which to sample. We call these HPO-procedure-input values *hyper-hyperparameters* (hyper-HPs).[1] To make HPO outputs comparable, we also introduce the notion of a *log*:

Definition 1. *A log ℓ records all the choices and measurements made during an HPO*

[1]We provide a glossary of all definitions and symbols for reference at the beginning of Appendix B

run, including the total time T it took to run. It has all necessary information to make the HPO run reproducible.

A log can be thought of as everything needed to produce a table in a research paper: code, random seed, choice of hyper-HPs, information about the learning task, properties of the learning algorithm, all of the observable results. We formalize all of the randomness in HPO in terms of a random seed r and a pseudo-random number generator (PRNG) G. Given a seed, G deterministically produces a sequence of pseudo-random numbers: all numbers lie in some set $\mathcal{I}$ (typically 64-bit integers), i.e. $r \in \mathcal{I}$ and PRNG $G : \mathcal{I} \to \mathcal{I}^\infty$. With this, we can now define HPO formally:

Definition 2. *An HPO procedure H is a tuple $(H_*, \mathcal{C}, \Lambda, \mathcal{A}, \mathcal{M}, G, X)$ where H_* is a randomized algorithm, $\mathcal{C}$ is a set of allowable hyper-HPs (i.e., allowable configurations for H_*), Λ is a set of allowable HPs (i.e., of HP sets λ), $\mathcal{A}$ is a training algorithm (e.g. SGD), $\mathcal{M}$ is a model (e.g. VGG16), G is a PRNG, and X is some dataset (usually split into train and validation sets). When run, H_* takes as input a hyper-HP configuration $c \in \mathcal{C}$ and a random seed $r \in \mathcal{I}$, then proceeds to run $\mathcal{A}_\lambda$ (on $\mathcal{M}_\lambda$ using $G(r)$ and data[2] from X) some number of times for different HPs $\lambda \in \Lambda$. Finally, H_* outputs a tuple (λ^*, ℓ), where λ^* is the HP configuration chosen by HPO and ℓ is the log documenting the run.*

Running H is a crucial part of model development. As part of an empirical, scientific procedure, we specify different training algorithms and a learning task, run potentially many HPO passes, and try to make general conclusions about overall algorithm performance. That is, we aim to develop knowledge

[2]Definition 2 does not preclude cross-validation, as this can be part of H_*. The input dataset X can be split in various ways, as a function of the random seed r.

regarding whether one of the algorithms outperforms the others. However, recent empirical findings indicate that it is actually really challenging to pick hyper-HPs that yield reliable knowledge about general algorithm performance. In fact, it is a surprisingly common occurrence to be able to draw inconsistent conclusions based on our choice of hyper-HPs [122, 176, 376, 543].

An example illustrating the possibility of drawing inconsistent conclusions from HPO. As a first step to studying HPO as a procedure for developing reliable knowledge, we provide an example of how being inadvertently deceived by HPO is a real problem, even in excellent research (we give an additional example in Appendix B).[3] We first reproduce Wilson et al. [623], in which the authors trained VGG16 with different optimizers on CIFAR-10 (Figure 2.1a). This experiment uses grid search, with a powers-of-2 grid for the learning rate α crossed with the default HPs for Adam. Based on the best-performing HPO per algorithm ($\alpha = 1$), it is reasonable to conclude that non-adaptive methods (e.g., SGD) perform better than adaptive ones (e.g., Adam [314]), as the non-adaptive optimizers demonstrate higher test accuracy.

However, this setting of grid search's hyper-HPs directly informs this particular conclusion; using different hyper-HPs makes it possible to conclude the opposite. Inspired by Choi et al. [122], we perform grid search over a different subspace, tuning both learning rate and Adam's ϵ parameter. Our results entail the logically opposite conclusion: Non-adaptive methods *do not* outperform adaptive ones. Rather, when choosing the HPs that maximize test accuracy, all of the optimizers essentially have equivalent performance (Figure 2.1b, Appendix B). Notably, as we can see from the confidence intervals in Figure 2.1, **satisfying statistical significance is not sufficient to avoid being deceived**

[3] All code can be found at `https://github.com/pasta41/deception`.

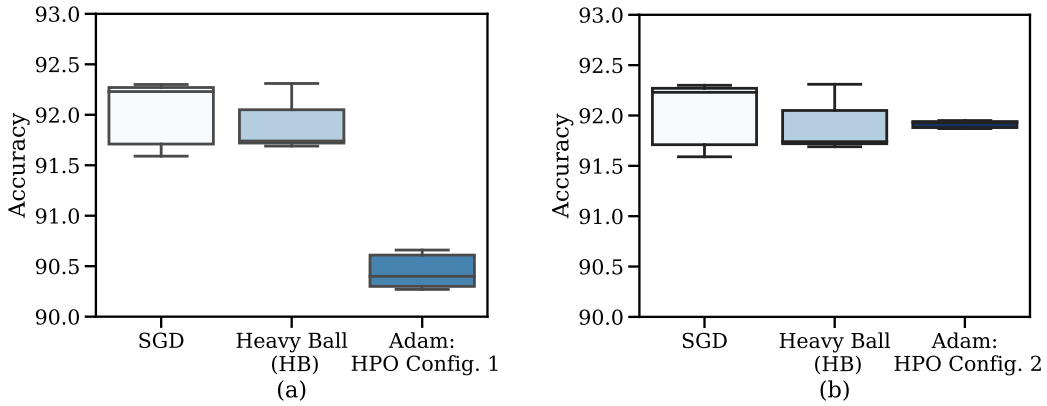


Figure 2.1: Demonstrating the possibility of drawing inconsistent conclusions from HPO (what we shorthand *hyperparameter deception*) when training VGG16 on CIFAR-10. Each box plot represents a log. In (a), we replicate Wilson et al. [623] and show the best-performing results: One can reasonably conclude that Adam under-performs non-adaptive methods. In (b), we change the HPO search space for Adam, and similarly show the best-performing results: In contradiction, one can reasonably conclude that Adam performs just as well as non-adaptive methods in terms of test accuracy.

about comparative algorithm performance [632]. Thus, we will require additional tools aside from statistical tests to reason about this, which we discuss in Sections 2.4 & 2.5.

This example is not exceptional, or even particularly remarkable, in terms of illustrating the hyperparameter deception problem. We simply chose it for convenience: The experiment does not require highly-specialized ML sub-domain expertise to understand it, and it is arguably broadly familiar, as it very well-cited [623]. However, we emphasize that hyperparameter deception is rather common. Additional examples can be found in numerous empirical studies across ML subfields [80, 122, 176, 387, 403, 430, 521, 543] (Appendix B). This work shows that reported results tend to be impressive for the tested hyper-HP configurations, but that modifying HPO can lead to vastly different performance outcomes that entail contradictory conclusions.

More generally, it is possible to develop results that are wrong about performance, or else correct about performance but for the wrong reasons (e.g., by picking "lucky" hyperparameters). Neither of these outcomes constitutes reliable knowledge [232, 355]. As scientists, this is disheartening. We want to have confidence in the conclusions we draw from our experiments. We want to trust that we are deriving reliable knowledge about algorithm performance. **In the sections that follow, our aim is to study HPO in this reliable-knowledge sense: We want to develop ways to reason rigorously and confidently about how we derive knowledge from empirical investigations involving HPO.**

2.3 Epistemic Hyperparameter Optimization

Our discussion in Section 2.2 shows that applying standard HPO methodologies can be deceptive: Our beliefs about algorithm performance can be controlled by happenstance, wishful thinking, or, even worse, potentially by an adversary trying to trick us with a tampered set of HPO logs. This leaves us in a position where the "knowledge" we derived may not be knowledge at all—since we could have easily (had circumstances been different) concluded the opposite. To address this, we propose that the process of drawing conclusions using HPO should itself be an object of study. We formalize this reasoning process, which we call *epistemic hyperparameter optimization* (EHPO), and we provide an intuition for how EHPO can help us think about the hyperparameter deception problem.

Definition 3. *An **epistemic hyperparameter optimization procedure (EHPO)** is a tuple $(\mathcal{H}, \mathcal{F})$ where $\mathcal{H}$ is a set of HPO procedures H (Definition 2) and $\mathcal{F}$ is a function that maps a set of HPO logs $\mathcal{L}$ (Definition 1) to a set of logical formulas $\mathcal{P}$, i.e. $\mathcal{F}(\mathcal{L}) =$*

$\mathcal{P}$. An execution of EHPO involves running each $H \in \mathcal{H}$ some number of times (each run produces a log ℓ), and then evaluating $\mathcal{F}$ on the logs $\mathcal{L}$ produced in order to output the conclusions $\mathcal{F}(\mathcal{L})$ we draw from all of the HPO runs.

In practice, it is common to run EHPO for two training algorithms, $\mathcal{J}$ and $\mathcal{K}$, and to compare their performance to conclude which is better-suited for the task at hand. $\mathcal{H}$ contains at least one HPO that runs $\mathcal{J}$ and at least one HPO that runs $\mathcal{K}$. The possible conclusions in output $\mathcal{P}$ include p = "$\mathcal{J}$ performs better than $\mathcal{K}$", and $\neg p$ = "$\mathcal{J}$ does not perform better than $\mathcal{K}$". Intuitively, EHPO is deceptive whenever it could produce p and also could (if configured differently or due to randomness) produce $\neg p$. That is, we can be deceived if the EHPO procedure we use to derive knowledge about algorithm performance could entail logically inconsistent results.

Our example in Section 2.2 is deceptive because using different hyper-HP-configured grid searches for $\mathcal{H}$ could produce contradictory conclusions. We ran two variants of EHPO $(\mathcal{H}, \mathcal{F})$: The first replicated Wilson et al. [623]'s original $\mathcal{H}$ of 3 grid-searches on SGD, HB, and Adam (Figure 2.1a), and the second used 3 grid-searches with a modified grid search for Adam that also tuned ϵ (Figure 2.1b). Each EHPO produced a $\mathcal{L}$ with 3 logs. For both, to draw conclusions $\mathcal{F}$ picks the best-performing HP-config per $\mathcal{A}$ and maps them to formulas including "SGD outperforms Adam." From the 3 logs in Figure 2.1a, we conclude p: "Non-adaptive optimizers outperform adaptive ones"; from the 3 logs in Figure 2.1b, we conclude $\neg p$: "Non-adaptive methods do not outperform adaptive ones." How can we formally reason about EHPO to avoid this possibility of drawing inconsistent conclusions—to guard against deceiving ourselves about algorithm performance when running EHPO?

Framing an adversary who can deceive us. To begin answering this question, we take inspiration from Descartes' deceptive demon thought experiment (Appendix B). We frame the problem in terms of a powerful adversary trying to deceive us—one that can cause us to doubt ourselves and our conclusions. Notably, the demon is not a real adversary; rather, it models a worst-case setting of configurations and randomness that are usually set arbitrarily or by happenstance in EHPO.

Imagine an evil demon who is trying to deceive us about the relative performance of different algorithms via running EHPO. At any time, the demon maintains a set $\mathcal{L}$ of HPO logs, which it can modify either by running an HPO $H \in \mathcal{H}$ with whatever hyper-HPs $c \in C$ and seed $r \in \mathcal{I}$ it wants (producing a new log ℓ, which it adds to $\mathcal{L}$) or by erasing some of the logs in its set. Eventually, it stops and presents us with $\mathcal{L}$, from which we will draw some conclusions using $\mathcal{F}$, i.e. $\mathcal{F}(\mathcal{L})$.

The demon's EHPO could deceive us via the conclusions we draw from the set of logs it produces. For example, $\mathcal{L}$ may lead us to conclude that one algorithm performs better than another, when in fact picking a different set of hyper-HPs could have generated logs that would lead us to conclude differently. We want to be sure that we will not be deceived by any logs the demon ***could*** produce. Of course, this intuitive definition is lacking: It is not clear what is meant by ***could***. Our contribution in the sections that follow is to pin down a formal, reasonable definition of ***could*** in this context, so that we can suggest an EHPO procedure that can defend against such a maximally powerful adversary. We intentionally imagine such a powerful adversary because, if we can defend against it, then we will also be defended against weaker or accidental

deception.

2.4 A Logic for Reasoning about EHPO

The informal notion of ***could*** established above encompasses numerous sources of uncertainty. There is the time to run EHPO and the choices of random seed, algorithms to compare, HPO procedures, hyper-HPs, and learning task. Then, once we have completed EHPO and have a set of logs, we have to digest those logs into logical formulas from which we base our conclusions. This introduces more uncertainty, as we need to reason about whether we believe those conclusions or not.

Our formalization needs to capture all of these sources of uncertainty, and needs to be sufficiently expressive to capture how they could combine to cause us to believe deceptive conclusions. It needs to be expansive enough to handle the common case — of a well-intentioned researcher with limited resources making potentially incorrect conclusions — and the rarer, worst case — of gaming results.

Why not statistics? As the common toolkit in ML, statistics might seem like the right choice for modeling all this uncertainty. However, statistics is great for reasoning about uncertainty that is *quantifiable*. For this problem, not all of the sources of uncertainty are easily quantifiable. In particular, it is very difficult to quantify the different hyper-HP possibilities. It is not reasonable to model hyper-HP selection as a random process; we do not sample from a distribution and, even if we wanted to, it is not clear how we would pick the distribution

from which to sample. Moreover, as we saw in our example in Section 2.2, testing for statistical significance is not sufficient to prevent deception. While the results under consideration may be statistically significant, they can still fail to prevent the possibility of yielding inconsistent conclusions. For this reason, when it comes to deception, statistical significance can even give us false confidence in the conclusions we draw.

Why modal logic? Modal logic is the standard mathematical tool for formalizing reasoning about uncertainty [118, 190] — for formalizing the thus far informal notion of what the demon ***could*** bring about running EHPO. It is meant precisely for dealing with different types of uncertainty, particularly uncertainty that is difficult to quantify, and has been successfully employed for decades in AI [70, 255, 256], programming languages [132, 341, 472], and distributed systems [207, 265, 457]. In each of these computer science fields, modal logic's flexible semantics has been indispensable for writing proofs about higher-level specifications with multiple sources of not-precisely-quantifiable, lower-level uncertainty.

For example, in distributed computing, it lets us write proofs about overall system correctness, abstracting away from the specific non-determinism introduced by each lower-level computing process [207]. Analogously, modal logic can capture the uncertainty in EHPO without being prescriptive about particular hyper-HP choices. Our notion of correctness, which we want to reason about and guarantee, is not being deceived. Therefore, while modal logic may be an atypical choice for ML, it comes with a huge payoff. By constructing the right semantics, we can capture all the sources of uncertainty described above and we can write simple proofs about whether we can be deceived by the EHPO we

run. In Section 2.5, it is this formalization that ultimately enables us to naturally suggest a defense against being deceived.

2.4.1 Introducing our logic: syntax and semantics overview

Modal logic inherits the tools of more-familiar propositional logic and adds two operators: $\Diamond$ to represent *possibility* and $\Box$ to represent *necessity*. These operators enable reasoning about *possible worlds*—a semantics for representing how the world *is* or *could be*, making modal logic the natural choice to express the "could" intuition from Section 2.3. The well-formed formulas ϕ of modal logic are given recursively in Backus-Naur form, where P is any atomic proposition:

$$\phi := P \mid \neg\phi \mid \phi \wedge \phi \mid \Diamond\phi$$

$\Diamond p$ reads, "It is possible that p."; p is true at *some* possible world, which we *could* reach (Appendix B). Note that $\Box$ is syntactic sugar, with $\Box p \equiv \neg\Diamond\neg p$. Similarly, "or" has $p \vee q \equiv \neg(\neg p \wedge \neg q)$ and "implies" has $p \rightarrow q \equiv \neg p \vee q$. The axioms of modal logic are as follows:

$$\vdash Q \rightarrow \Box Q \quad \textit{(necessitation)}. \qquad \Box(Q \rightarrow R) \rightarrow (\Box Q \rightarrow \Box R) \quad \textit{(distribution)}.$$

where Q and R are any formula, and $\vdash Q$ means Q is a theorem of propositional logic. We can now provide the syntax and an intuitive notion of the semantics of our logic for reasoning about deception.

Syntax. Our logic requires an extension of standard modal logic. We need *two* modal operators to reckon with two overarching modalities: the possible results of the demon running EHPO ($\Diamond_t$) and our beliefs about conclusions from those

results ($\mathcal{B}$). Combining these modalities yields well-formed formulas ψ where, for any atomic proposition P and any positive real t,

$$\psi := P \mid \neg\psi \mid \psi \wedge \psi \mid \Diamond_t \psi \mid \mathcal{B}\psi$$

Note the EHPO modal operator here is *indexed*: $\Diamond_t$ captures "how possible" ($\Diamond$) something is, quantified by the compute capabilities of the demon (t) [70, 190, 267].

Semantics intuition. We suppose that an EHPO user has in mind some atomic propositions (propositions of the background logic unrelated to possibility or belief, such as "the best-performing log for $\mathcal{J}$ has lower loss than the best-performing log for $\mathcal{K}$") with semantics that are already defined. $\wedge$ and $\neg$ inherit their semantics from ordinary propositional logic, which can combine propositions to form formulas. A set of EHPO logs $\mathcal{L}$ (Definition 1) can be digested into such logical formulas. That is, we define our semantics using logs $\mathcal{L}$ as models over formulas p: $\mathcal{L} \models p$, which reads "$\mathcal{L}$ models p", means that p is true for the set of logs $\mathcal{L}$. We will extend this intuition to give semantics for possibility $\Diamond_t$ (Section 2.4.2) and belief $\mathcal{B}$ (Section 2.4.3), culminating in a tool that lets us reason about whether or not EHPO can deceive us by possibly yielding inconsistent conclusions (Section 2.4.4).

Using our concrete example to ground us. To clarify our presentation below, we will map our semantics to the example from Section 2.2, providing an informal intuition before formal definitions.

2.4.2 Expressing the possible outcomes of EHPO using $\Diamond_t$

Our formalization for possible EHPO is based on the demon of Section 2.3. Recall, the demon models a worst-case scenario. In practice, we deal with the easier case of well-intentioned ML researchers. The notion of possibility we define here gives limits on what possible world a demon *with bounded EHPO time* could reliably bring about. We first define a *strategy* the demon can execute for EHPO:

Definition 4. *A randomized* ***strategy*** $\boldsymbol{\sigma}$ *is a function that specifies which action the demon will take. Given* $\mathcal{L}$*, its current set of logs,* $\boldsymbol{\sigma(\mathcal{L})}$ *gives a distribution over concrete actions, where each action is either 1) running a new* H *with its choice of hyper-HPs* c *and seed* r *2) erasing some logs, or 3) returning. We let* Σ *denote the set of all such strategies.*

The demon we model controls the hyper-HPs c and the random seed r, but importantly does *not* fully control the PRNG G. From the adversary's perspective, for a strategy σ to be reliable it must succeed regardless of the specific G. Informally, the demon cannot hack the PRNG.[4]

Informally, we now want to *execute a strategy* to bring about a particular outcome p. In Section 2.2, our good-faith strategy was simple: We ran each H with its own hyper-HPs and random seed, then returned. The demon is trickier: It is adopting a strategy to try to bring about a deceptive outcome. **Formally**, we model the demon executing strategy σ on logs $\mathcal{L}$ with a PRNG unknown to the demon as follows. Let $\mathcal{G}$ denote the distribution over PRNGs $G : \mathcal{I} \to \mathcal{I}^\infty$, in

[4]We do not consider adversaries that can directly control how data is ordered and submitted to the algorithms under evaluation. This distinction shows that our logical construction non-trivial: We are able to defend against strong adversaries that can game the output of EHPO, which is separate from cheating by hacking the PRNG.

which all number sequence elements are drawn independently and uniformly from $\mathcal{I}$ (recall, $\mathcal{I}$ is typically the 64-bit integers). First, draw G from $\mathcal{G}$, conditioned on G being consistent with all the runs in $\mathcal{L}$.[5] The demon then performs a random action drawn from $\sigma(\mathcal{L})$, using G as the PRNG when running a new HPO H, and continues—updating the working set of logs $\mathcal{L}$ as it goes—until the "return" action is chosen.

Using this process, we define what outcomes p the demon can reliably bring about (i.e., what is possible, $\Diamond$) in the EHPO output logs $\mathcal{L}$ by running this random strategy σ in bounded time t. **Informally**, $\Diamond_t p$ means that an adversary could adopt a strategy σ that is guaranteed to cause the desired outcome p to be the case while taking time at most t in expectation. In Section 2.2, where p is "Non-adaptive methods outperform adaptive ones", Figure 2.1a shows $\Diamond_t p$. **Formally**,

Definition 5. *Let* $\boldsymbol{\sigma[\mathcal{L}]}$ *denote the logs output from executing strategy* σ *on logs* $\mathcal{L}$*, and let* $\boldsymbol{\tau_\sigma(\mathcal{L})}$ *denote the total time spent during execution.* $\tau_\sigma(\mathcal{L})$ *is equivalent to the sum of the times* T *it took each HPO procedure* $H \in \mathcal{H}$ *executed in strategy* σ *to run. Note that both* $\sigma[\mathcal{L}]$ *and* $\tau_\sigma(\mathcal{L})$ *are random variables, as a function of the randomness of selecting* G *and the actions sampled from* $\sigma(\mathcal{L})$*. For any formula* p *and any* $t \in \mathbb{R}_{>0}$*, we say* $\mathcal{L} \models \Diamond_t p$*, i.e. "*$\mathcal{L}$ *models that it is possible* p *in time* t*," if*

$$\textit{there exists a strategy } \sigma \in \Sigma, \textit{ such that } \quad \mathbb{P}(\sigma[\mathcal{L}] \models p) = 1 \;\; \textit{and} \;\; \mathbb{E}[\tau_\sigma(\mathcal{L})] \leq t.$$

We will usually choose t to be an upper bound on what is considered a reasonable amount of time to run EHPO. It does not make sense for t to be unbounded, since this corresponds to the unrealistic setting of having infinite com-

[5] i.e., All random events recorded in $\mathcal{L}$ should agree with the corresponding random numbers produced by G.

pute time to perform HPO runs. We model our budget in terms of time; however, we could use this setup to reason about other monotonically increasing resource costs, such as energy usage. Our indexed modal logic inherits many axioms of modal logic, with indexes added (Appendix B), e.g.:

$$\vdash (p \to q) \to (\Diamond_t p \to \Diamond_t q) \quad \textit{(necess. + distribution)} \qquad p \to \Diamond_t p \quad \textit{(reflexivity)}$$

$$\Diamond_t \Diamond_s p \to \Diamond_{t+s} p \quad \textit{(transitivity)} \qquad \Diamond_s \Box_t p \to \Box_t p \quad \textit{(symmetry)}$$

$$\Diamond_t (p \wedge q) \to (\Diamond_t p \wedge \Diamond_t q) \quad \textit{(dist. over } \wedge\textit{)},$$

To summarize: The demon knows all possible hyper-HPs; it can pick whichever ones it wants to run EHPO within a bounded time budget t to realize the outcome p it wants. That is, if with some probability the demon can deceive us in some amount of time, then the demon can reliably deceive us with any larger time budget: If the demon fails to produce a deceptive result, it can use the strategy of just re-running until it yields the result it desires. Since $\Diamond_t$ models the worst-case all-powerful demon, it can also model any weaker EHPO user with time budget t.

2.4.3 Expressing how we draw conclusions using $\mathcal{B}$

We employ the modal operator $\mathcal{B}$ from the logic of belief[6] to model ourselves as an observer who believes in the truth of the conclusions drawn from running EHPO. $\mathcal{B}p$ reads "It is concluded that p." For example, when comparing the performance of two algorithms for a task, p could be "$\mathcal{J}$ is better than $\mathcal{K}$" and thus $\mathcal{B}p$ would be understood as, "It is concluded that $\mathcal{J}$ is better than $\mathcal{K}$."

[6] $\mathcal{B}$ is syntactically analogous to the $\Box$ modal operator in standard modal logic [277, 527, 590] (Appendix B).

We model ourselves as a consistent *Type 1* reasoner [550]. **Informally**, this means we believe all propositional tautologies (necessitation), our belief distributes over implication (distribution), and we do not derive contradictions (consistency). We do not require completeness: We allow the possibility of not concluding anything about p (i.e., neither $\mathcal{B}p$ nor $\mathcal{B}\neg p$). **Formally**, for any formulas p and q,

$$\vdash p \rightarrow \mathcal{B}p \quad \textit{(necess.)}; \quad \mathcal{B}(p \rightarrow q) \rightarrow (\mathcal{B}p \rightarrow \mathcal{B}q) \quad \textit{(dist.)}; \quad \neg(\mathcal{B}p \wedge \mathcal{B}\neg p) \quad \textit{(consistency)}.$$

To understand our belief semantics, recall that EHPO includes a function $\mathcal{F}$, which maps a set of output logs $\mathcal{L}$ to our conclusions (i.e., $\mathcal{F}(\mathcal{L}) = \mathcal{P}$ is our set of conclusions). **Informally**, when our conclusion set $\mathcal{F}(\mathcal{L})$ contains a formula p, we say the set of logs $\mathcal{L}$ models our belief $\mathcal{B}$ in that formula p. In Section 2.2, the logs of Figure 2.1a model $\mathcal{B}p$ and the logs of Figure 2.1b model $\mathcal{B}\neg p$. **Formally**,

Definition 6. *For any formula p, we say $\mathcal{L} \models \mathcal{B}p$, "$\mathcal{L}$ models our belief in p", if $p \in \mathcal{F}(\mathcal{L})$.*

Note we constrain what $\mathcal{F}$ can output. For a reasonable notion of belief, $\mathcal{F}$ must model the consistent *Type 1* reasoner axioms above. Otherwise, deception aside, $\mathcal{F}$ is an unreasonable way to draw conclusions, since it is not even compatible with our belief logic.

2.4.4 Expressing hyperparameter deception

So far we have defined the semantics of our two separate modal operators, $\Diamond_t$ and $\mathcal{B}$. We now begin to reveal the benefit of using modal logic for our formalization. These operators can interact to formally express what we informally

illustrated in Section 2.2: a notion of hyperparameter deception. It is a well-known result that we can combine modal logics [525] (Appendix B). We do so to define an axiom that, if satisfied, guarantees EHPO will not be able to deceive us. For any formula p,

$$\neg\left(\Diamond_t \mathcal{B}p \ \wedge \ \Diamond_t \mathcal{B}\neg p\right) \qquad \textit{(t-non-deceptive)}.$$

Informally, our running example can be considered a proof by exhibition: It violates this axiom because Figure 2.1a's logs model $\Diamond_t \mathcal{B}p$ and Figure 2.1b's logs model $\Diamond_t \mathcal{B}\neg p$. That is, $\Diamond_t \mathcal{B}p \wedge \Diamond_t \mathcal{B}\neg p$ using grid search for this task.

For the worst-case, *t-non-deceptiveness* expresses the following: **If there exists a strategy σ by which the demon could get us to conclude p in t expected time, then there can exist no t-time strategy by which the demon could have gotten us to believe $\neg p$.** To make this concrete, suppose our t-non-deceptive axiom holds for an EHPO method that results in p. Intuitively, given a maximum reasonable time budget t, if there is no adversary that can consistently control whether we believe p or its negation when running that EHPO, then the EHPO is defended against deception. Conversely, if an adversary could consistently control our conclusions, then the EHPO is potentially gameable. That is, if our t-non-deceptive axiom does not hold (i.e., we can be deceived, $\Diamond_t \mathcal{B}p \wedge \Diamond_t \mathcal{B}\neg p$), then even if we conclude p after running EHPO, we cannot claim to *know* p. Our belief as to the truth-value of p could be under the complete control of an adversary—or just a result of happenstance.

To summarize: An EHPO is t-non-deceptive if it satisfies all of the axioms above. Our example in Section 2.2 is t-deceptive because the axioms do not hold. The semantics of these axioms capture all of the possible uncertainty from

the process of drawing conclusions from EHPO–and how that uncertainty can combine to cause us to believe t-deceptive conclusions.

2.5 Constructing Defended EHPO

Now that we have a formal notion of what it means for EHPO to be (non)-deceptive, **we can write proofs about what it means for an EHPO method to be guaranteed to be deception-free**. Importantly, these proofs will increase our confidence that our conclusions from EHPO are not due to the happenstance of picking a particular set of hyper-HPs.

To talk about defenses, we need to understand what it means to construct a "defended reasoner." In other words, for an EHPO $(\mathcal{H}, \mathcal{F})$, we need $\mathcal{F}$ to yield conclusions that we can defend against deception. Recall from Definition 6 that logs $\mathcal{L}$ model our belief in a formula p, i.e. $\mathcal{L} \models \mathcal{B}p \equiv p \in \mathcal{F}(\mathcal{L})$. With this in mind, we begin by supposing we have a naive EHPO $(\mathcal{H}, \mathcal{F}_{\text{n}})$ featuring a naive reasoner $\mathcal{B}_{\text{n}}$ with corresponding belief function $\mathcal{F}_{\text{n}}$. We want to construct a new "defended reasoner" $\mathcal{B}_*$ that has a "skeptical" belief function $\mathcal{F}_*$. $\mathcal{F}_*$ should weaken the conclusions of $\mathcal{F}_{\text{n}}$ (i.e., $\mathcal{F}_*(\mathcal{L}) \subseteq \mathcal{F}_{\text{n}}(\mathcal{L})$ for any $\mathcal{L}$) and result in an EHPO $(\mathcal{H}, \mathcal{F}_*)$ that is guaranteed to be t-non-deceptive. In other words, defended reasoner $\mathcal{B}_*$ never concludes more than the naive reasoner $\mathcal{B}_{\text{n}}$. **Informally**, a straightforward way to do this is to have $\mathcal{B}_*$ conclude p only if both the naive $\mathcal{B}_{\text{n}}$ would have concluded p, and it is impossible for an adversary to get $\mathcal{B}_{\text{n}}$ to conclude $\neg p$ in time t. **Formally**, construct $\mathcal{B}_*$ such that for any p,

$$\mathcal{B}_* p \equiv \mathcal{B}_{\text{n}} p \wedge \neg \Diamond_t \mathcal{B}_{\text{n}} \neg p \tag{2.1}$$

Directly from our axioms (Section 2.4), we can now prove $\mathcal{B}_*$ is defended. We will suppose it is possible for $\mathcal{B}_*$ to be deceived, demonstrate a contradiction, and thereby guarantee that $\mathcal{B}_*$ is t-non-deceptive. Suppose $\mathcal{B}_*$ can be deceived in time t, i.e. $\Diamond_t \mathcal{B}_* p \wedge \Diamond_t \mathcal{B}_* \neg p$ is True. Starting with the left, $\Diamond_t \mathcal{B}_* p$:

			Rule
$\Diamond_t \mathcal{B}_* p$	$\equiv$	$\Diamond_t (\mathcal{B}_n p \wedge \neg \Diamond_t \mathcal{B}_n \neg p)$	Applying $\Diamond_t$ to the definition of $\mathcal{B}_* p$ (2.1)
	$\rightarrow$	$\Diamond_t (\neg \Diamond_t \mathcal{B}_n \neg p)$	Reducing a conjunction to either of its terms: $(a \wedge b) \rightarrow b$
	$\rightarrow$	$\neg \Diamond_t \mathcal{B}_n \neg p$	Symmetry; dropping all but the right-most operator: $\Diamond_t (\Diamond_t a) \rightarrow \Diamond_t a$

We then pause to apply our axioms to the right side of the conjunction, $\Diamond_t \mathcal{B}_* \neg p$:

			Rule
$\Diamond_t \mathcal{B}_* \neg p$	$\equiv$	$\Diamond_t (\mathcal{B}_n \neg p \wedge \neg \Diamond_t \mathcal{B}_n p)$	Applying $\Diamond_t$ to the definition of $\mathcal{B}_* \neg p$ (2.1)
	$\rightarrow$	$\Diamond_t \mathcal{B}_n \neg p \wedge \Diamond_t \neg \Diamond_t \mathcal{B}_n p$	Distributing $\Diamond_t$ over $\wedge$: $\Diamond_t (a \wedge b) \rightarrow (\Diamond_t a \wedge \Diamond_t b)$
	$\rightarrow$	$\Diamond_t \mathcal{B}_n \neg p$	Reducing a conjunction to either of its terms: $(a \wedge b) \rightarrow a$

We now bring both sides of the conjunction back together: $\Diamond_t \mathcal{B}_* p \wedge \Diamond_t \mathcal{B}_* \neg p \equiv \neg \Diamond_t \mathcal{B}_n \neg p \wedge \Diamond_t \mathcal{B}_n \neg p$. The right-hand side is of the form $\neg a \wedge a$, which must be False. This contradicts our initial assumption that $\mathcal{B}_*$ is t-deceptive (i.e., $\Diamond_t \mathcal{B}_* p \wedge \Diamond_t \mathcal{B}_* \neg p$ is True). Therefore, $\mathcal{B}_*$ is t-non-deceptive.

This example illustrates the power of our choice of formalization. In just a few lines of simple logic, we can validate defenses against deception. **This analysis shows that a t-defended reasoner $\mathcal{B}_*$ *is always possible*,** and it does so without needing to refer to the particular underlying semantics of an EHPO. However, we intend this example to only be illustrative, as it may not be practical to compute $\mathcal{B}_*$ as defined in (1) if we cannot easily evaluate whether $\Diamond_t \mathcal{B}_n \neg p$. We next suggest a concrete EHPO with a defended $\mathcal{B}_*$, and show how deception

can be avoided in our Section 2.2 example by using this EHPO instead of grid search.

A defended random search EHPO. Random search takes two hyper-HPs, a distribution μ over the HP space and a number of trials $K \in \mathbb{N}$ to run. HPO consists of K independent trials of training algorithms $\mathcal{A}_{\lambda_1}, \mathcal{A}_{\lambda_2}, \ldots, \mathcal{A}_{\lambda_K}$, where the HPs λ_k are independently drawn from μ, taking expected time proportional to K. When drawing conclusions, we usually look at the "best" run for each algorithm. For simplicity, we suppose there is only one algorithm, $\mathcal{A}$. We bound how much the choice of hyper-HPs can affect the HPs, and define a defended EHPO based on a variant of random search.

Definition 7. *Suppose that we are given a naive EHPO procedure $(\{H\}, \mathcal{F}_n)$, in which H is random search and is the only HPO in our EHPO, and $\mathcal{F}_n$ is a "naive" belief function associated with a naive reasoner $\mathcal{B}_n$. For any $K, R \in \mathbb{N}$, we define the "(K,R)-defended" belief function $\mathcal{F}_*$ for a skeptical reasoner $\mathcal{B}_*$ as the following conclusion-drawing procedure. First, $\mathcal{F}_*$ only makes conclusion set $\mathcal{P}_*$ from a single log $\hat{\ell}$ with $K*R$ trials; otherwise, it concludes nothing, outputting $\emptyset$. Second, $\mathcal{F}_*$ splits the single $\hat{\ell}$ into R logs $\ell_1, \ell_2, \ldots, \ell_R$, each containing K independent-random-search trials.*[7] *Finally, $\mathcal{F}_*$ outputs the intersection of what the naive reasoner would have output on each log ℓ_i,*

$$\mathcal{F}_*(\{\hat{\ell}\}) = \mathcal{P}_* \equiv \mathcal{F}_n(\{\ell_1\}) \cap \mathcal{F}_n(\{\ell_2\}) \cap \cdots \cap \mathcal{F}_n(\{\ell_R\}).$$

Equivalently, $\{\hat{\ell}\} \models \mathcal{B}_* p$ only if $\{\ell_i\} \models \mathcal{B}_n p$ for all i.

Informally, to draw a conclusion using this EHPO, $\mathcal{B}_*$ splits a random-search-trial log of size $K * R$ into R groups of K-trial logs, passing each K-trial log to one of an ensemble of R naive reasoners $\mathcal{B}_{\text{n}}$. $\mathcal{B}_*$ only concludes p if all

[7] This is not generally allowable. $\mathcal{F}_*$ can do this because random-search logs contain interchangeable trials.

R naive reasoners unanimously agree on p. We can guarantee this EHPO to be t-non-deceptive by assuming a bound on how much the hyper-HPs can affect the HPs.

Theorem 1. *Suppose that the set of allowable hyper-HPs C of H is constrained, such that any two allowable random-search distributions μ and ν have Renyi-∞-divergence at most a constant, i.e. $D_\infty(\mu||\nu) \leq \gamma$. The (K, R)-defended random-search EHPO of Definition 7 is guaranteed to be t-non-deceptive if we set $R \geq \sqrt{t \exp(\gamma K)/K} = O(\sqrt{t})$.*

We prove Theorem 1 in the Appendix B. This result shows that our defense is *actually* a defense, and moreover it defends with a log size $K * R$—and compute requirement for good-faith EHPO—that scales sublinearly in t. A good-faith actor can, in sublinear-in-t time, produce a log (of length $K * R$) that will allow our t-non-deceptive reasoner to reach conclusions. This means that we defend against adversaries with much larger compute budgets than are expected from good-faith actors.

Validating our defense empirically and selecting hyper-HPs. Any defense ultimately depends on the hyper-HPs it uses. Thus, we should have a reasonable belief that choosing differently would not have led an opposite conclusion. We therefore run a two-phased search [122, 271, 496], repeating our VGG16-CIFAR10 experiment from Section 2.2. First, we run a coarse-grained, dynamic protocol to find reasonable hyper-HPs for Adam's ϵ; second, we use those hyper-HPs to run our defended random search. We start with a distribution to search over ϵ, and note that the performance is best on the high end. We change the hyper-HPs, shifting the distribution until Adam's performance starts to degrade, and use the resulting hyper-HPs ($\epsilon \in [10^{10}, 10^{12}]$) to run our defense (Appendix B).

Algorithm 1 Defense with Random Search

Require: Set of $K * R$ random-search logs $\{\mathcal{L}_i\}_{i=1}^{KR}$, defense subsampling budget M, criterion constant δ, subsample size κ

for $m = 1, \cdots, M$ **do**
    Subsample κ logs: $\{\mathcal{L}_i\}_{i=1}^{\kappa} \sim \{\mathcal{L}_i\}_{i=1}^{KR}$.
    Obtain conclusions $\{\mathcal{P}_i\}_{i=1}^{\kappa}$ from $\{\mathcal{L}_i\}_{i=1}^{\kappa}$.
    Obtain output conclusion for m: $\mathcal{P}^{(m)} \leftarrow \text{Majority}(\{\mathcal{P}_i\}_{i=1}^{\kappa})$
end for
if $\exists p$ s.t. $\geq (1-\delta)M$ of $\{\mathcal{P}^{(m)}\}_{i=1}^{M}$ conclude p **then**
    Conclude p.
else
    Conclude nothing.
end if

Table 2.1: Results from repeating our Section 2.2 experiment, using Algorithm 1 instead of grid search. p = "Non-adaptive optimizers (SGD and Heavy Ball) perform better than the adaptive optimizer Adam".

	p	$\neg p$	$1-\delta$	Conclude
SGD vs. Adam	0.213	0.788	0.75	$\neg p$
			0.8	Nothing
			0.9	Nothing
Heavy Ball vs. Adam	0.168	0.832	0.75	$\neg p$
			0.8	$\neg p$
			0.9	Nothing

We now run a modified version of our defended EHPO in Definition 7, described in Algorithm 1, with $K * R = 600$ (200 logs for each optimizer). Using a budget of $M = 10000$ iterations, we subsample $\kappa = 11$ logs and pass them to an ensemble of κ naive reasoners $\mathcal{B}_{\text{n}}$. We use κ logs, relaxing the requirement of using all $K * R$ logs in Definition 7, for efficiency. Each iteration m concludes the majority conclusion of the κ-sized $\mathcal{B}_{\text{n}}$ ensemble. This is why we set κ to an

odd number—to avoid ties. $\mathcal{B}_*$ draws conclusions based on the results of the M-majority conclusions. That is, we further relax the requirements of Definition 7: Instead of requiring unanimity, $\mathcal{B}_*$ only requires agreement on the truth-value of p for a fractional subset of M. We set this fraction using parameter $\delta \in [0, 1]$, where δ controls how skeptical our defended reasoner $\mathcal{B}_*$ is (lower δ corresponding to more skepticism). $\mathcal{B}_*$ concludes p when at least $(1-\delta)$ of our M subsampled runs concluded p. When this threshold is not met, $\mathcal{B}_*$ remains skeptical and concludes nothing. We summarize our final results in Table 2.1, and provide complete results in the Appendix B. Given how similar the optimizers all perform on this task (similar to Figure 2.1), being more skeptical increases the likelihood that we do not conclude anything.

2.6 Conclusion and Practical Takeaways

Much recent empirical work illustrates that it is easy to draw inconsistent conclusions from HPO [80, 122, 176, 387, 403, 430, 521, 543]. We call this problem *hyperparameter deception* and, to derive a defense, **argue that the process of drawing conclusions using HPO should itself be an object of study**. Taking inspiration from Descartes' demon, we formalize a logic for studying an epistemic HPO procedure. The demon can run any number of reproducible HPO passes to try to get us to believe a particular notion about algorithm performance. Our formalization enables us to not believe deceptive notions: It naturally suggests how to guarantee that an EHPO is defended against deception. We offer recommendations to avoid hyperparameter deception in practice (we expand on this in Appendix B):

- **Researchers should construct their own notion of skepticism $\mathcal{B}_*$, appropriate to their specific task.** There is no one-size-fits-all defense solution. Our results are ***broad insights*** about defended EHPO: A defended EHPO is ***always possible***, but finding an efficient one will depend on the task.
- **Researchers should make explicit how they choose hyper-HPs.** What is reasonable is ultimately a function of what the ML community accepts. Being explicit, rather than eliding hyper-HP choices, is essential for helping decide what is reasonable. As a heuristic, we recommend setting hyper-HPs such that they include HPs for which the optimizers' performance starts to degrade, as we do above.
- **Avoiding hyperparameter deception is just as important as reproducibility**. We have shown that reproducibility [80, 249, 271, 470, 542] is only part of the story for ensuring reliability. While necessary for guarding against brittle findings, it is not sufficient. We can replicate results—even statistically significant ones—that suggest conclusions that are altogether wrong.

More generally, our work is a call to researchers to reason more rigorously about their beliefs concerning algorithm performance. In relation to EHPO, this is akin to challenging researchers to reify their notion of $\mathcal{B}$—to justify their belief in their conclusions from the HPO. Such epistemic rigor concerning drawing conclusions from empirical studies has a long history in more mature branches of science and computing, including evolutionary biology [243], statistics [227, 228], programming languages [431], and computer systems [215] (Appendix B). We believe that applying similar rigor will contribute significantly to the ongoing effort of making ML more robust and reliable.

CHAPTER 3

ARBITRARINESS AND SOCIAL PREDICTION

We next discuss a different source of arbitrariness in machine learning: how stochasticity based on the particular training-data examples can result in wildly variable classification outcomes in algorithmic fairness contexts. The prior chapter (Chapter 2) used predominantly theoretical tools to formally characterize a particular type of arbitrariness due to non-deterministic human decisions. Here, our methods are almost entirely experimentally driven. They yield important insights that we have begun to translate to law and policy (Chapter 4).

Chapter summary: Variance in predictions across different trained models is a significant, under-explored source of error in fair binary classification. In practice, the variance on some data examples is so large that decisions can be effectively *arbitrary*. To investigate this problem, we take an experimental approach and make four overarching contributions. We define a metric called *self-consistency*, derived from variance, which we use as a proxy for measuring and reducing arbitrariness. We then develop an ensembling algorithm that abstains from classification when a prediction would be arbitrary, and conduct a large-scale experimental study of the role of variance (*vis-a-vis* self-consistency and arbitrariness) in fair binary classification. Our experiments reveal shocking insights about the reliability of conclusions on benchmark datasets.

This chapter is a licensed derivative copy of work published and awarded Best Student Paper (Honorable Mention) at *AAAI 2024* [141]. This work grew out of oral-awarded *AIES 2021* [136] and *ICLR 2021* workshop [210] papers. Cooper and Abrams [136] deals with conceptual mismeasurement in ML in relation to construct validity; this chapter, in contrast, focuses on arbitrariness in experi-

mental measurements. This work was also greatly influenced by earlier research on scalable uncertainty estimation (Chapter 5 and Zhang et al. [642]).

3.1 Introduction

A goal of algorithmic fairness is to develop techniques that measure and mitigate discrimination in automated decision-making. In fair binary classification, this often involves training a model to satisfy a chosen *fairness metric*, which typically defines fairness as parity between model error rates for different demographic groups in the dataset [45]. However, even if a model's classifications satisfy a particular fairness metric, it is not necessarily the case that the model is equally confident in each classification.

To provide an intuition for what we mean by confidence, consider the following experiment: we fit 100 logistic regression models using the same learning process, which draws different subsamples of the training set from the `COMPAS` prison recidivism dataset [213, 344], and we compare the resulting classifications for two individuals in the test set. Figure 3.1 shows a difference in the consistency of predictions for both individuals: the 100 models agree completely to classify Individual 1 as "will recidivate" and disagree completely on whether to classify Individual 2 as "will" or "will not recidivate."

If we were to pick one model at random to use in practice, there would be no effect on how Individual 1 is classified; yet, for Individual 2, the prediction is effectively random. We can interpret this disagreement to mean that the learning process that produced these predictions is not sufficiently confident to justify assigning Individual 2 *either decision outcome*. In practice, instances

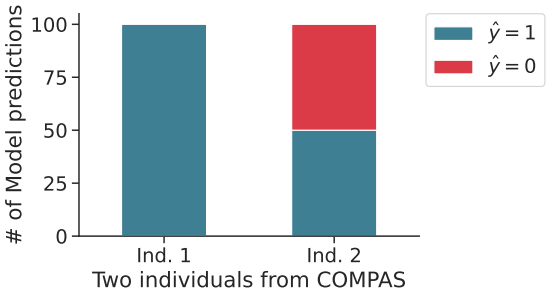


Figure 3.1: 100 bootstrapped logistic regression models show models can be very consistent in predictions $\hat{y}$ for some individuals (Ind. 1) and arbitrary for others (Ind. 2).

like Individual 2 exhibit so little confidence that their classification is effectively *arbitrary* [138, 146, 161]. Further, this arbitrariness can also bring about discrimination if classification decisions are *systematically more arbitrary* for individuals in certain demographic groups.

A key aspect of this example is that we use only one model to make predictions. This is the typical setup in fair binary classification: popular metrics are commonly applied to evaluate the fairness of a *single model* [260, 318, 471]. However, as is clear from the example learning process in Fig. 3.1, using only a single model can mask the arbitrariness of predictions. Instead, to reveal arbitrariness, we must examine *distributions over possible models for a given learning process*. With this shift in frame, we ask: *What is the empirical role of arbitrariness in fair binary classification tasks?*

To study this question, we:

1. **Quantify arbitrariness.** We formalize a metric called *self-consistency*, derived from statistical variance, which we use as a quantitative proxy for arbitrariness of model outputs. Self-consistency is a simple yet powerful tool for empirical analyses of fair classification (Section 3.3).

2. **Ensemble to improve self-consistency.** We extend Breiman's classic bagging algorithm [84] to allow for abstaining from classifying instances for which self-consistency is low. This improves overall self-consistency (i.e., reduces variance), and improves accuracy (Section 3.4).

3. **Perform a comprehensive experimental study of variance in fair binary classification.** We conduct the largest-to-date such study, through the lens of self-consistency and its relationship to arbitrariness. Surprisingly, we find that the benchmarks we evaluate are *close-to-fair* when taking into account the amount of arbitrariness present in predictions — *before* we even try to apply *any* fairness interventions (Section 3.5). This finding casts doubt on the reliability of prior work that uses individual models to make claims that there is baseline unfairness in these benchmarks (Section 3.6).

3.2 Preliminaries on Fair Binary Classification

To analyze arbitrariness in the context of fair binary classification, we first need to establish our background definitions. This material is likely familiar to most readers. Nevertheless, we highlight particular details that are important for understanding the experimental methods that enable our contributions. We present the fair-binary-classification problem formulation and associated empirical approximations, with an emphasis on the *distribution over possible models* that could be produced from training on different subsets of data drawn from the same data distribution.

Problem formulation. Consider a distribution $q(\cdot)$ from which we can sample *examples* $(\boldsymbol{x}, \boldsymbol{g}, o)$. The $\boldsymbol{x} \in \mathbb{X} \subseteq \mathbb{R}^m$ are feature *instances* and $\boldsymbol{g} \in \mathbb{G}$ is a group of *protected attributes* that we do not use for learning (e.g., race, gender).[1] The $o \in \mathbb{O}$ are the associated *observed labels*, and $\mathbb{O} \subseteq \mathbb{Y}$, where $\mathbb{Y} = \{0, 1\}$ is the label space. From $q(\cdot)$ we can sample training datasets $\{(\boldsymbol{x}, \boldsymbol{g}, o)\}_{i=1}^{n}$, with $\mathbb{D}$ representing the set of all n-sized datasets. To reason about the possible models of a hypothesis class $\mathbb{H}$ that could be learned from the different subsampled datasets $\boldsymbol{D}_k \in \mathbb{D}$, we define a *learning process*:

Definition 8. *A **learning process** is a randomized function that runs instances of a **training procedure** $\mathcal{A}$ on each $\boldsymbol{D}_k \in \mathbb{D}$ and a model specification, in order to produce **classifiers** $h_{\boldsymbol{D}_k} \in \mathbb{H}$. A particular run $\mathcal{A}(\boldsymbol{D}_k) \to h_{\boldsymbol{D}_k}$, where $h_{\boldsymbol{D}_k} : \mathbb{X} \to \mathbb{Y}$, which is deterministic mapping from the instance space $\mathbb{X}$ to the label space $\mathbb{Y}$. All such runs over $\mathbb{D}$ produce a distribution over possible trained models, μ.*

Reasoning about μ, rather than individual models $h_{\boldsymbol{D}_k}$, enables us to contextualize arbitrariness in the data, which, in turn, is captured by learned models (Section 3.3).[2] Each particular model $h_{\boldsymbol{D}_k} \sim \mu$ deterministically produces classifications $\hat{y} = h_{\boldsymbol{D}_k}(\boldsymbol{x})$. The classification rule is $h_{\boldsymbol{D}_k}(\boldsymbol{x}) = \mathbf{1}[r_{\boldsymbol{D}_k}(\boldsymbol{x}) \geq \tau]$, for some threshold τ, where regressor $r_{\boldsymbol{D}_k} : \mathbb{X} \to [0, 1]$ computes the probability of positive classification. Executing $\mathcal{A}(\boldsymbol{D}_k)$ produces $h_{\boldsymbol{D}_k} \sim \mu$ by minimizing the *loss* of predictions $\hat{y}$ with respect to their associated observed labels o in $\boldsymbol{D}_k$. This loss is computed by a chosen *loss function* $\ell : \mathbb{Y} \times \mathbb{Y} \mapsto \mathbb{R}$. We compute predictions for a *test set* of fresh examples and calculate their loss. The loss is an estimate of the *error* of $h_{\boldsymbol{D}_k}$, which is dependent on the specific

[1] We examine the common setting in which $|\boldsymbol{g}| = 1$, and abuse notation, treating $\boldsymbol{g}$ like a scalar with $\mathbb{G} = \{0, 1\}$.

[2] Model multiplicity has similar aims, but ultimately relocates the arbitrariness we describe to model selection (Section 3.6 and Appendix C.3.3).

dataset $\boldsymbol{D}_k$ used for training. To generalize to the error of all possible models produced by a specific learning process (Definition 8), we consider the *expected error*, $\texttt{Err}(\mathcal{A}, \mathbb{D}, (\boldsymbol{x},\ \boldsymbol{g},\ o)) = \mathbb{E}_{\mathbf{D}}[\ell(o,\ \hat{y})|\mathbf{x} = \boldsymbol{x}]$.

In fair classification, it is common to use *0-1 loss* $\triangleq \mathbf{1}[\hat{y} \neq o]$ or *cost-sensitive loss*, which assigns asymmetric costs C_{01} for false positives $\texttt{FP}$ and C_{10} for false negatives $\texttt{FN}$ [188]. These costs are related to the classifier threshold $\tau = \frac{C_{01}}{C_{01}+C_{10}}$, with $C_{01}, C_{10} \in \mathbb{R}^+$ (Appendix C.1.3). Common fairness metrics, such as Equality of Opportunity [260], further analyze error by computing disparities across group-specific error rates $\texttt{FPR}_{\boldsymbol{g}}$ and $\texttt{FNR}_{\boldsymbol{g}}$. For example, $\texttt{FPR}_{\boldsymbol{g}} \triangleq p_\mu[r_{\mathbf{D}}(\mathbf{x}) \geq \tau | o = 0, \mathbf{g} = \boldsymbol{g}] = p_\mu[\hat{y} = 1 | o = 0, \mathbf{g} = \boldsymbol{g}]$. Model-specific $\texttt{FPR}_{\boldsymbol{g}}$ and $\texttt{FNR}_{\boldsymbol{g}}$ are further-conditioned on the dataset used in training, i.e., $\mathbf{D} = \boldsymbol{D}_k$.

Empirical approximation of the problem formulation. We typically only have access to one dataset, not the data distribution $q(\cdot)$. In fair binary classification experiments, it is common to estimate expected error by performing *cross validation* (CV) on this dataset to produce a small handful of models [119, 154]. CV can be unreliable when there is high variance; it can produce error estimates that are themselves high variance, and does not reliably estimate expected error with respect to possible models μ (Section 3.5). For more details, see Efron [184] and Efron and Tibshirani [186] and Wager [608].

To get around these reliability issues, one can *bootstrap*.[3] Bootstrapping splits the available data into train and test sets, and simulates drawing different training datasets from a distribution by resampling the train set $\hat{\boldsymbol{D}}$, generating replicates $\hat{\boldsymbol{D}}_1, \hat{\boldsymbol{D}}_2, \ldots, \hat{\boldsymbol{D}}_B \coloneqq \hat{\mathbb{D}}$. We use these replicates $\hat{\mathbb{D}}$ to approximate the learning

[3]We could use MCMC [641, 642], but optimization is the standard tool that allows use of standard models in fairness.

process on $\mathbb{D}$ (Def. 8). We treat the resulting $\hat{h}_{\hat{\boldsymbol{D}}_1}, \hat{h}_{\hat{\boldsymbol{D}}_2}, \ldots, \hat{h}_{\hat{\boldsymbol{D}}_B}$ as our empirical estimate for the distribution $\hat{\mu}$, and evaluate their predictions for the *same reserved test set*. This enables us to produce comparisons of classifications across test instances like in Figure 3.1 (Appendix C.1.4).

3.3 Variance, Self-Consistency and Arbitrariness

We develop a quantitative proxy for measuring arbitrariness, called *self-consistency* (Section 3.3.2), which is derived from a definition of statistical *variance* between different model predictions (Section 3.3.1). We then illustrate how self-consistency is a simple-yet-powerful tool for revealing the role of arbitrariness in fair classification (Section 3.3.3). Next, we will introduce an algorithm to improve self-consistency (Section 3.4) and compute self-consistency on popular fair binary classification benchmarks (Section 3.5).

3.3.1 Arbitrariness Resembles Statistical Variance

In Section 3.2, we discussed how common fairness definitions analyze error by computing false positive rate (`FPR`) and false negative rate (`FNR`). Another common way to formalize error is as a decomposition of different statistical sources: *noise*-, *bias*-, and *variance*-induced error [7, 229]. To understand our metric for self-consistency (Section 3.3.2), we first describe how the arbitrariness in Figure 3.1 (almost, but not quite) resembles variance.

Informally, variance-induced error quantifies fluctuations in individual example predictions for different models $h_{\boldsymbol{D}_k} \sim \mu$. Variance is the error in the learn-

ing process that comes from training on different datasets $\boldsymbol{D}_k \in \mathbb{D}$. In theory, we measure variance by imagining training all possible $h_{\boldsymbol{D}_k} \sim \mu$, testing them all on the same test instance $(\boldsymbol{x}, \boldsymbol{g})$, and then quantifying how much the resulting classifications for $(\boldsymbol{x}, \boldsymbol{g})$ deviate *from each other*. More formally,

Definition 9. *For all pairs of possible models $h_{\boldsymbol{D}_i}, h_{\boldsymbol{D}_j} \sim \mu$ $(i \neq j)$, the **variance** for a test $(\boldsymbol{x}, \boldsymbol{g})$ is*

$$\texttt{var}(\mathcal{A}, \mathbb{D}, (\boldsymbol{x}, \boldsymbol{g})) \triangleq \mathbb{E}_{h_{\boldsymbol{D}_i} \sim \mu, h_{\boldsymbol{D}_j} \sim \mu}\Big[\ell\Big(h_{\boldsymbol{D}_i}(\boldsymbol{x}), h_{\boldsymbol{D}_j}(\boldsymbol{x})\Big)\Big].$$

We can approximate variance directly by using the bootstrap method (Section 3.2, Appendices C.1.4 and C.2.1). For 0-1 and cost-sensitive loss with costs $C_{01}, C_{10} \in \mathbb{R}^+$ (Section 3.2), we can generate B replicates to train B concrete models that serve as our approximation for the distribution $\hat{\mu}$. For $B = B_0 + B_1 > 1$, where B_0 and B_1 denote the number of 0- and 1-class predictions for $(\boldsymbol{x}, \boldsymbol{g})$,

$$\begin{aligned} \hat{\texttt{var}}(\mathcal{A}, \hat{\mathbb{D}}, (\boldsymbol{x}, \boldsymbol{g})) &\coloneqq \frac{1}{B(B-1)} \sum_{i \neq j} \ell\Big(\hat{h}_{\hat{\boldsymbol{D}}_i}(\boldsymbol{x}), \hat{h}_{\hat{\boldsymbol{D}}_j}(\boldsymbol{x})\Big) \\ &= \frac{(C_{01} + C_{10}) B_0 B_1}{B(B-1)}. \end{aligned} \tag{3.1}$$

We derive (3.1) in Appendix C.2.2 and show that, for increasingly large B, $\hat{\texttt{var}}$ is defined on $[0, \frac{C_{01}+C_{10}}{4} + \epsilon]$.

3.3.2 Defining Self-Consistency from Variance

It is clear from above that, in general, variance (3.1) is unbounded. We can always increase the maximum possible $\hat{\texttt{var}}$ by increasing the magnitudes of our chosen C_{01} and C_{10} (Section 3.2).[4] However, as we can see from our intu-

[4] Because $\tau = \frac{C_{01}}{C_{01}+C_{10}}$, for a given τ we can scale costs arbitrarily and have the same decision rule Relative, not absolute, costs affect the number of classifications B_0 and B_1.

ition for arbitrariness in Figure 3.1, the most important takeaway is the amount of (dis)agreement, reflected in the counts B_0 and B_1. Here, there is no notion of the cost of misclassifications. So, variance (3.1) does not exactly measure what we want to capture. Instead, we want to focus unambiguously on the (dis)agreement part of variance, which we call *self-consistency of the learning process*:

Definition 10. *For all pairs of possible models $h_{\boldsymbol{D}_i}, h_{\boldsymbol{D}_j} \sim \mu$ $(i \neq j)$, the **self-consistency of the learning process** for a test $(\boldsymbol{x}, \boldsymbol{g})$ is*

$$\begin{aligned} \mathtt{SC}(\mathcal{A}, \mathbb{D}, (\boldsymbol{x}, \boldsymbol{g})) &\triangleq \mathbb{E}_{h_{\boldsymbol{D}_i} \sim \mu, h_{\boldsymbol{D}_j} \sim \mu}\Big[h_{\boldsymbol{D}_i}(\boldsymbol{x}) = h_{\boldsymbol{D}_j}(\boldsymbol{x})\Big] \\ &= p_{h_{\boldsymbol{D}_i} \sim \mu, h_{\boldsymbol{D}_j} \sim \mu}(h_{\boldsymbol{D}_i}(\boldsymbol{x}) = h_{\boldsymbol{D}_j}(\boldsymbol{x})). \end{aligned} \tag{3.2}$$

In words, (3.2) models the probability that two models produced by the same learning process on different n-sized training datasets agree on their predictions for the same test instance. Like variance, we can derive an empirical approximation of $\mathtt{SC}$. Using the bootstrap method with $B = B_0 + B_1 > 1$,

$$\begin{aligned} \hat{\mathtt{SC}}(\mathcal{A}, \hat{\mathbb{D}}, (\boldsymbol{x}, \boldsymbol{g})) &\coloneqq \frac{1}{B(B-1)} \sum_{i \neq j} \mathbf{1}\Big[\hat{h}_{\hat{\boldsymbol{D}}_i}(\boldsymbol{x}) = \hat{h}_{\hat{\boldsymbol{D}}_j}(\boldsymbol{x})\Big] \\ &= 1 - \frac{2B_0B_1}{B(B-1)}. \end{aligned} \tag{3.3}$$

For increasingly large B, $\hat{\mathtt{SC}}$ is defined on $[0.5 - \epsilon, 1]$ (Appendix C.2.3). Throughout, we use the shorthand *self-consistency*, but it is important to note that Definition 10 is a property of the distribution over possible models μ produced by the learning process, not of individual models. We summarize other important takeaways below:

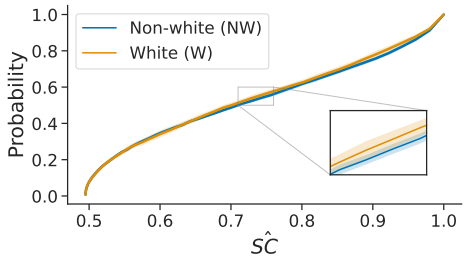
Terminology. In naming our metric, we intentionally evoke related notions of "consistency" in logic and the law [218, 560] (Appendix C.2.4).

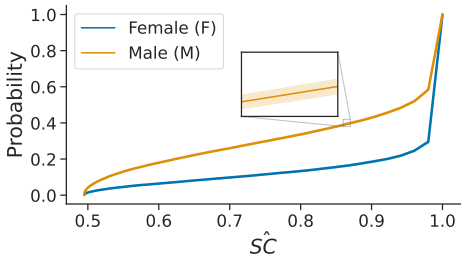
Interpretation. Definition 10 is defined on [0.5, 1], which coheres with the intuition in Figure 3.1: 0.5 and 1 respectively reflect minimal (Individual 2) and maximal (Individual 1) possible SC. SC, unlike FPR and FNR (Section 3.2), does *not* depend on the observed label o. It captures the learning process's confidence in a classification $\hat{y}$, but says nothing directly about $\hat{y}$'s accuracy. By construction, low self-consistency indicates high variance, and vice versa. We derive empirical $\widehat{\text{SC}}$ (3.3) from $\widehat{\text{var}}$ (3.1) by leveraging observations about the definition of $\widehat{\text{var}}$ for 0-1 loss (Appendix C.2.3). While there are no costs C_{01}, C_{10} in computing (3.3), they still affect empirical measurements of $\widehat{\text{SC}}$. Because C_{01} and C_{10} affect τ (Section 3.2), they control the concrete number of B_0 and B_1, and thus the $\widehat{\text{SC}}$ we measure in experiments.

Empirical focus. Since self-consistency depends on the particular data subsets used in training, conclusions about its relevance vary according to task. This is why we take a practical approach for our main results — of running a large-scale experimental study on many different datasets to extract general observations about $\widehat{\text{SC}}$'s practical effects (Section 3.5). In our experiments, we typically use $B = 101$, which yields a $\widehat{\text{SC}}$ range of $[\approx 0.495, 1]$ in practice.[5]

Relationship to other fairness concepts. Self-consistency is qualitatively different from traditional fairness metrics. Unlike FPR and FNR, SC does not depend on observed label o. This has two important implications. First, while calibration also measures a notion of confidence, it is different: calibration reflects confidence with respect to *a model* predicting o, but says nothing about the relative confidence in predictions $\hat{y}$ produced by the *possible models* μ that result

[5] Efron and Tibshirani [186] recommend $B \in \{50 \dots 200\}$.





	$\Delta\hat{\mathbf{Err}}$	$\Delta\hat{\mathbf{FPR}}$	$\Delta\hat{\mathbf{FNR}}$
	1.0 ± 1.4%	2.0 ± 1.4%	0.9 ± 1.4%
	$\hat{\mathbf{Err}}$	$\hat{\mathbf{FPR}}$	$\hat{\mathbf{FNR}}$
Total	36.6 ± 0.5%	17.3 ± 0.8%	19.3 ± 0.7%
NW	36.9 ± 0.5%	18.0 ± 0.7%	19.0 ± 0.8%
W	35.9 ± 1.3%	16.0 ± 1.2%	19.9 ± 1.1%

(a) `COMPAS` split by `race`; random forests (RFs)

	$\Delta\hat{\mathbf{Err}}$	$\Delta\hat{\mathbf{FPR}}$	$\Delta\hat{\mathbf{FNR}}$
	12.2 ± 0.4%	6.0 ± 0.3%	6.3 ± 0.3%
	$\hat{\mathbf{Err}}$	$\hat{\mathbf{FPR}}$	$\hat{\mathbf{FNR}}$
Total	17.3 ± 0.3%	7.7 ± 0.3%	9.6 ± 0.1%
F	9.0 ± 0.3%	3.7 ± 0.1%	5.3 ± 0.3%
M	21.2 ± 0.3%	9.7 ± 0.3%	11.6 ± 0.1%

(b) `Old Adult` split by `sex`; random forests (RFs)

Figure 3.2: $\widehat{\texttt{SC}}$ CDFs for `COMPAS` (3.2a) and `Old Adult` (3.2b). We train random forests (B = 101 replicates), and repeat with 10 train/test splits to produce (very tight) confidence intervals. $\widehat{\texttt{SC}}$ is effectively identical across subgroups $\boldsymbol{g}$ in `COMPAS`; `Old Adult` exhibits systematic differences in arbitrariness across $\boldsymbol{g}$. Tables show mean ± STD of the relative disparities, e.g., $\Delta\hat{\texttt{Err}} = |\hat{\texttt{Err}}_0 - \hat{\texttt{Err}}_1|$ (top); and, the absolute $\hat{\texttt{Err}}$, $\hat{\texttt{FPR}}$, $\hat{\texttt{FNR}}$, and $\widehat{\texttt{SC}}$, also broken down by $\boldsymbol{g}$ (bottom).

from the learning process [471]. Second, a common assumption in algorithmic fairness is that there is *label bias* — that unfairness is due in part to discrimination reflected in recorded, observed decisions o [136, 212]. As a result, it is arguably a nice side effect that self-consistency does not depend on o. However, it is also possible to be perfectly self-consistent and inaccurate (e.g., $\forall k, \hat{y}_k \neq o$; Section 3.6).

3.3.3 Illustrating Self-Consistency in Practice

$\widehat{\texttt{SC}}$ enables us to evaluate arbitrariness in classification experiments. It is straightforward to compute $\widehat{\texttt{SC}}$ (3.3) with respect to multiple test instances $(\boldsymbol{x}, \boldsymbol{g})$ — for all instances in a test set or for all instances conditioned on membership in $\boldsymbol{g}$. Therefore, beyond visualizing $\widehat{\texttt{SC}}$ for individuals (Figure 3.1), we can also do so across sets of individuals.

We plot the cumulative distribution (CDF) of $\widehat{\texttt{SC}}$ for the groups $\boldsymbol{g}$ in the test set (i.e., the x-axis shows the range of $\widehat{\texttt{SC}}$ for $B = 101$, $[\approx 0.495, 1]$). In Figure 3.2, we provide illustrative examples from two of the most common fair classification benchmarks [194], `COMPAS` and `Old Adult` using random forests (RFs). We split the available data into train and test sets, and bootstrap the train set $B = 101$ times to train models $\hat{h_1}, \hat{h_2}, \ldots, \hat{h_{101}}$ (Section 3.2). We repeat this process on 10 train/test splits, and the resulting confidence intervals (shown in the inset) indicate that our $\widehat{\texttt{SC}}$ estimates are stable. We group observations into two categories:

Individual arbitrariness. Both CDFs show that $\widehat{\texttt{SC}}$ varies drastically across test instances. For random forests on the `COMPAS` dataset, about one-half of instances are under .7 self-consistent. Nearly one-quarter of test instances are effectively .5 self-consistent; they resemble Individual 2 in Figure 3.1, meaning that their predictions are essentially arbitrary. These differences in $\widehat{\texttt{SC}}$ across the test set persist even though the 101 models exhibit relatively small average disparities $\Delta\widehat{\texttt{Err}}$, $\Delta\widehat{\texttt{FPR}}$, and $\Delta\widehat{\texttt{FNR}}$ (Figure 3.2a, bottom; Section 3.5.2). This supports our motivating claim: it is possible to come close to satisfying fairness metrics, while the learning process exhibits very different levels of confidence

for the underlying classifications that inform those metrics (Section 3.1).

Systematic arbitrariness. We can also highlight $\widehat{\texttt{SC}}$ according to groups $\boldsymbol{g}$. The $\widehat{\texttt{SC}}$ plot for `Old Adult` shows that it is possible for the degree of arbitrariness to be *systematically worse* for a particular demographic $\boldsymbol{g}$ (Figure 3.2b). While the lack of $\widehat{\texttt{SC}}$ is not as extreme as it is for `COMPAS` (Figure 3.2a) — the majority of test instances exhibit over .9 $\widehat{\texttt{SC}}$ — there is more arbitrariness in the `Male` subgroup. We can quantify such *systematic arbitrariness* using a measure of distance between probability distributions. We use the Wasserstein-1 distance ($\mathcal{W}_1$), which has a closed form for CDFs [489]. The $\mathcal{W}_1$ distance has an intuitive interpretation for measuring systematic arbitrariness: it computes the total disparity in `SC` by examining all possible `SC` levels κ at once (Appendix C.2.3). For two groups $\boldsymbol{g} = 0$ and $\boldsymbol{g} = 1$ with respective `SC` CDFs F_0 and F_1, $\mathcal{W}_1 \triangleq \int_{\mathbb{R}} |F_0(\kappa) - F_1(\kappa)| \, d\kappa$. For `Old Adult`, $\hat{\mathcal{W}}_1 = 0.127$; for `COMPAS`, which does not show systematic arbitrariness, $\hat{\mathcal{W}}_1 = 0.007$.

3.4 Accounting for Self-Consistency

By definition, low $\widehat{\texttt{SC}}$ signals that there is high $\hat{\texttt{var}}$ (Section 3.3.2). It is therefore a natural idea to use variance reduction techniques to improve $\widehat{\texttt{SC}}$ (and thus reduce arbitrariness).

As a starting point for improving $\widehat{\texttt{SC}}$, we perform variance reduction with Breiman's *bootstrap aggregation*, or *bagging*, ensembling algorithm [84]. Bagging involves bootstrapping to produce a set of B models (Section 3.2), and then, for each test instance, producing an aggregated prediction $\hat{y}_A$, which takes the ma-

jority vote of the $\hat{y}_1, \ldots, \hat{y}_B$ classifications. This procedure is practically effective for classifiers with high variance [84, 85]. However, by taking the majority vote, bagging embeds the idea that having slightly-better-than-random classifiers is sufficient for improving ensembled predictions, $\hat{y}_A$. Unfortunately, there exist instances like Individual 2 (Figure 3.1), where the classifiers in the ensemble are evenly split between classes. This means that bagging alone cannot overcome arbitrariness (Appendix C.4.1).

To remedy this, we add the option to abstain from prediction if $\hat{\texttt{SC}}$ is low (Algorithm 2). A minor adjustment to (3.3) accounts for abstentions, and a simple proof follows that Algorithm 2 improves $\hat{\texttt{SC}}$ (Appendix C.4). We bootstrap as usual, but produce a prediction $\hat{y} \in [0, 1]$ for instance $\boldsymbol{x}$ only if $\boldsymbol{x}$ surpasses a user-specified minimum level κ of $\hat{\texttt{SC}}$; otherwise, if an instance fails to achieve $\hat{\texttt{SC}}$ of at least κ, we `Abstain` from predicting. For evaluation, we divide the test set into two subsets: we group together the instances we `Abstain` on in an *abstention set* and those we predict on in a *prediction set*. This method improves self-consistency through two complementary mechanisms: 1) variance reduction (due to bagging, see Appendix C.4) and 2) abstaining from instances that exhibit low $\hat{\texttt{SC}}$ (thereby raising the overall amount of $\hat{\texttt{SC}}$ for the prediction set, see Appendix C.4).

Further, since variance is a component of error (Appendix 3.3), variance reduction also tends to improve accuracy [84]. This leads to an important observation: the abstention set, by definition, exhibits high variance; we can therefore expect it to exhibit higher error than the prediction set (Section 3.5). So, while at first glance it may seem odd that our solution for arbitrariness is to *not predict*, it is worth noting that we often would have predicted incorrectly on a large

Algorithm 2 $\widehat{\texttt{SC}}$ Ensembling with Abstention

Input: training dataset $(\boldsymbol{X}, \boldsymbol{o})$, $\mathcal{A}$, B, $\widehat{\texttt{SC}}$ $\kappa \in [0.5, 1]$, $\boldsymbol{x}_{\text{test}}$
Output: $\hat{y}$ with $\widehat{\texttt{SC}} \geq \kappa$ or `Abstain`

$\hat{y}_A$:= list() ▷ To store ensemble predictions
for $1 \ldots B$ **do**
  $\boldsymbol{D}_B \leftarrow$ Bootstrap($(\boldsymbol{X}, \boldsymbol{o})$)
  ▷ $\hat{h}_{\boldsymbol{D}_B}$ can itself be a bagged model, with $\mathcal{A}$ bagging on
    $\boldsymbol{D}_B$ as the dataset to bootstrap

  $\hat{h}_{\boldsymbol{D}_B} \leftarrow \mathcal{A}(\boldsymbol{D}_B)$
  $\hat{y}_A$.append($\hat{h}_{\boldsymbol{D}_B}(\boldsymbol{x}_{\text{test}})$) ▷ $\hat{y}_A = [\hat{y}_1, \ldots, \hat{y}_B]$
end for
return Aggregate($\hat{y}_A, \kappa$)
▷ Returns κ-majority prediction or abstains
function Aggregate($\hat{y}_1, \ldots, \hat{y}_B, \kappa$)
  if SelfConsistency($\hat{y}_1, \ldots, \hat{y}_B$) $\geq \kappa$ ▷ Compute $\widehat{\texttt{SC}}$ (3.3)
    return $\arg\max_{y' \in \mathbb{Y}} \left[\sum_{i=1}^{B} \mathbf{1}[y' = \hat{y}_i] \right]$
  end if
  return `Abstain`
end function

portion of the abstention set, anyway (Appendix C.4). In practice, we test two versions of our method:

Simple ensembling. We run Algorithm 2 to build ensembles of typical hypothesis classes in algorithmic fairness. For example, running with $B = 101$ decision trees and $\kappa = 0.75$ produces a bagged classifier that contains 101 underlying decision trees, for which the bagged classifier abstains from predicting on test instances that exhibit less than 0.75 $\widehat{\texttt{SC}}$. If overall $\widehat{\texttt{SC}}$ is low, then simple ensembling will lead to a large number of abstentions. For example, almost half of all test instances in `COMPAS` using random forests would fail to surpass the threshold $\kappa = 0.75$ (Figure 3.2a). The potential for large abstention sets informs our second approach.

Super ensembling. We run Algorithm 2 on *bagged* models $\hat{h}$. When there is low $\widehat{\texttt{SC}}$ (i.e., high $\texttt{vâr}$) it can be beneficial to do an initial pass of variance reduction. We produce bagged classifiers using traditional bagging, but without abstaining (at Algorithm 2, lines 4-5); *then* we Aggregate using those bagged classifiers as the underlying models $\hat{h}$. The first round of bagging raises the overall $\widehat{\texttt{SC}}$ before the second round, which is when we decide whether to `Abstain` or not. We therefore expect this approach to abstain less; however, it may potentially incur higher error, if, by happenstance, simple-majority-vote bagging chooses $\hat{y} \neq o$ for instances with very low $\widehat{\texttt{SC}}$ (Appendix C.4). We also experiment with an Aggregate rule that averages the output probabilities of the underlying regressors $r_{\boldsymbol{D}_k}$, and then applies threshold τ to produce ensembled predictions. We do not observe major differences in results.

3.5 Experiments

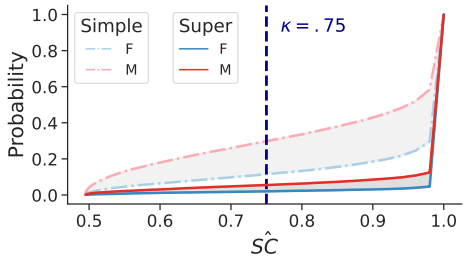
We release an extensible package of different Aggregate methods, with which we trained and compared several million different models (all told, taking on the order of 10 hours of compute). We include results covering common datasets and models: `COMPAS`, `Old Adult`, `German` and `Taiwan Credit`, and 3 large-scale `New Adult - CA` tasks on logistic regression (LR), decision trees (DTs), random forests (RFs), MLPs, and SVMs. By using Algorithm 2, we happened to observe close-to-fairness in nearly every task (Section 3.5.2).

Releasing an `HMDA` toolkit. A possible explanation is that most fairness benchmarks are small ($< 25,000$ examples) and therefore exhibit high variance. We

therefore clean a larger, more diverse, and newer dataset for investigating fair binary classification — the Home Mortgage Disclosure Act (HMDA) 2007-2017 datasets [206] — and release them with a standalone, easy-to-use software package. In this paper, we examine the NY and TX 2017 subsets of HMDA, which have 244,107 and 576,978 examples, respectively, and we still find close-to-fairness (Section 3.5.1).

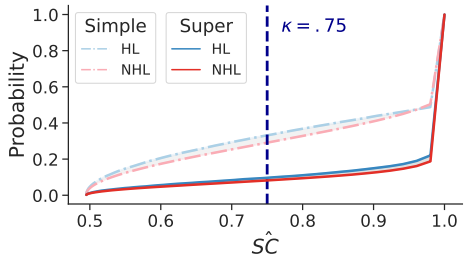
Presentation. To visualize Algorithm 2, we plot the CDFs of the $\widehat{\text{SC}}$ of the underlying models used in each ensembling method. We simultaneously plot the results of simple ensembling (dotted curves) and super ensembling (solid curves). Instances to the left of the vertical line (the minimum $\widehat{\text{SC}}$ threshold κ) form the abstention set. We also provide corresponding mean ± STD fairness and accuracy metrics for individual models (our expected, but not-necessarily-practically-attainable baseline) and for both simple and super ensembling. For ensembling methods, we report these metrics on the prediction set, along with the abstention rate ($\widehat{\text{AR}}$).

We necessarily defer most of our results to the online version of our paper. Here, we exemplify two overarching themes: the effectiveness of both ensembling variants (Section 3.5.1), and how our results reveal shocking insights about reliability in fair binary classification research (Section 3.5.2). For all experiments, we illustrate Algorithm 2 with $\kappa = 0.75$, but note that κ is task-dependent in practice.



	Baseline	Simple	Super
$\Delta\widehat{\text{FNR}}$	6.3 ± .3%	4.1 ± .3%	5.8 ± .4%
$\widehat{\text{FNR}}_{\text{F}}$	5.3 ± .3%	3.5 ± .1%	4.9 ± .2%
$\widehat{\text{FNR}}_{\text{M}}$	11.6 ± .1%	7.6 ± .3%	10.7 ± .3%

(a) `Old Adult` split by `sex`; random forests (RFs)



	Baseline	Simple	Super
$\Delta\widehat{\text{FNR}}$	0.7 ± .2%	1.1 ± .3%	2.2 ± .3%
$\widehat{\text{FNR}}_{\text{HL}}$	10.1 ± .2%	3.3 ± .3%	8.0 ± .3%
$\widehat{\text{FNR}}_{\text{NHL}}$	9.4 ± .1%	2.2 ± .1%	5.8 ± .1%

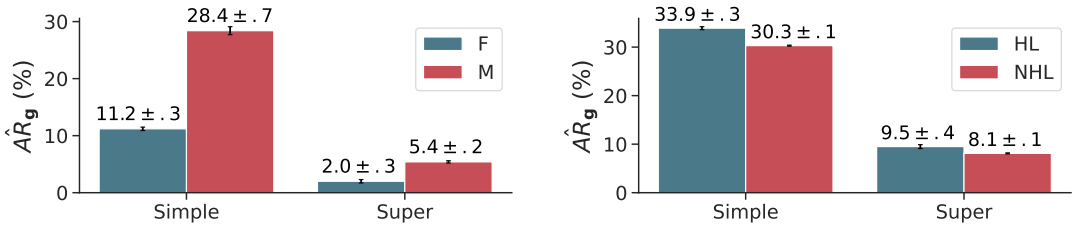
(b) `HMDA-NY-2017` split by `ethnicity`; random forests (RFs)

Figure 3.3: Algorithm 2: simple and super ensembling RFs for `Old Adult` (3.3a) and `HMDA-NY-2017` (3.3b). Tables show $\widehat{\text{FNR}}$ (mean ± STD) for individual models (Baseline) and each ensembling method's prediction set; B = 101, 10 train/test splits (Appendix E). To highlight systematic arbitrariness (Section 3.3.3), we shade in gray the area between group-specific $\widehat{\text{SC}}$ CDFs for each method. An initial pass of variance reduction in super significantly decreases the systematic arbitrariness in `Old Adult`.

3.5.1 Validating Algorithm 2

We highlight results for two illustrative examples: `Old Adult` and `HMDA-NY-2017`, for `ethnicity` (Hispanic or Latino (HL), Non-Hispanic or Latino (NHL)). We plot $\widehat{\text{SC}}$ CDFs and show $\widehat{\text{FNR}}$ metrics using random forests (RFs). For `Old Adult`, the expected disparity of the RF baseline is $\Delta\widehat{\text{FNR}} = 6.3\%$. The dashed set of curves plots the underlying $\widehat{\text{SC}}$ for these RFs (Figure 3.3a). When we apply simple to these RFs, overall $\hat{\text{Err}}$ decreases, shown in part by the decrease in $\widehat{\text{FNR}}_{\text{F}}$ and $\widehat{\text{FNR}}_{\text{M}}$. Fairness also improves: $\Delta\widehat{\text{FNR}}$ decreases to 4.1%. However, the corresponding $\widehat{\text{AR}}$ is quite high, especially for the `Male` subgroup ($\boldsymbol{g}$ = M, Figure 3.4).

As expected, super improves overall $\widehat{\text{SC}}$ through a first pass of variance re-



(a) `Old Adult`, $\boldsymbol{g}$ = `sex` (b) `HMDA-NY-2017`, $\boldsymbol{g}$ = `ethnicity`

Figure 3.4: Group-specific abstention rates $\hat{\text{AR}}_{\boldsymbol{g}}$. Super ensembling abstains less and more equally than simple ensembling.

duction (Section 3.4). The $\hat{\text{SC}}$ CDF curves are brought down, indicating a lower proportion of the test set exhibits low $\hat{\text{SC}}$. Abstention rate $\hat{\text{AR}}$ is lower and more equal (Figure 3.4); however, error, while still lower than the baseline RFs, has gone up for all metrics. There is also a decrease in systematic arbitrariness (Section 3.3.3): the dark gray area for super ($\hat{\mathcal{W}}_1 = .014$) is smaller than the light gray area for simple ($\hat{\mathcal{W}}_1 = .063$) (Appendix C.2.3).

For `HMDA` (Figure 3.3b), simple similarly improves $\hat{\text{FNR}}$, but has a less beneficial effect on fairness ($\Delta\hat{\text{FNR}}$). However, note that since the baseline is the empirical expected error over thousands of RF models, the specific $\Delta\hat{\text{FNR}}$ is not necessarily attainable by any individual model. In this respect, simple has the benefit of actually obtaining a specific (ensemble) model that yields this disparity reliably in practice: $\Delta\hat{\text{FNR}} = 1.1\%$ is the mean over 10 simple ensembles. Notably, this is extremely low, even without applying traditional fairness techniques. Similar to `Old Adult`, simple exhibits high $\hat{\text{AR}}$, which decreases with super at the cost of higher error. $\hat{\text{FNR}}$ still improves for both $\boldsymbol{g}$ in comparison to the baseline, but the benefits are unequally applied: $\hat{\text{FNR}}_{\text{W}}$ has a larger benefit, so $\Delta\hat{\text{FNR}}$ increases slightly.

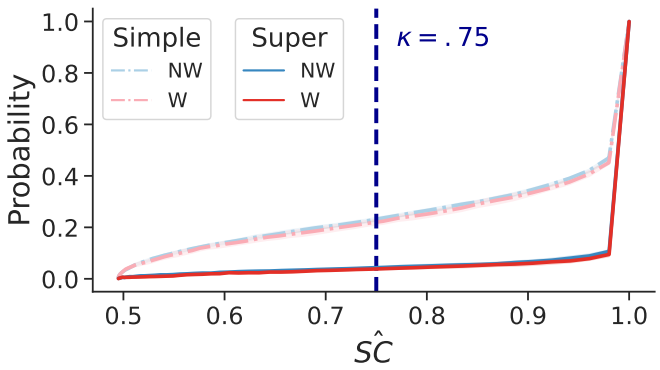
Abstention set error. As an example, the average total $\hat{\texttt{Err}}$ in the `Old Adult` simple abstention set is close to 40% — compared to 17% for the RF baseline, and 8% for simple and 14% for super prediction sets. As expected, beyond reducing arbitrariness, we abstain from predicting for many instances for which we also would have been more inaccurate (Section 3.4).

A trade-off. Our results support that there is indeed a trade-off between abstention rate and error (Section 3.4). This is because Algorithm 2 identifies low-$\hat{\texttt{SC}}$ instances for which ML prediction does a poor job, and abstains from predicting on them. Nevertheless, it may be infeasible for some applications to tolerate a high $\hat{\texttt{AR}}$. Thus the choice of κ and ensembling method should be considered a context-dependent decision.

Unequal abstention rates. When there is a high degree of systematic arbitrariness, $\hat{\texttt{AR}}$ can vary a lot by $\boldsymbol{g}$ (Figure 3.4). With respect to improving $\hat{\texttt{SC}}$, error, and fairness this may be a reasonable outcome: it is arguably better to abstain unevenly — deferring a final classification to non-ML decision processes — than to predict more inaccurately and arbitrarily for one group. More importantly, we rarely observe systematic arbitrariness; unequal $\hat{\texttt{AR}}$ is uncommon in practice (Section 3.6).

3.5.2 A Problem of Empirical Algorithmic Fairness

We also highlight results for `COMPAS`, 1 of the 3 most common fairness datasets [194]. Algorithm 2 is similarly very effective at reducing arbitrariness (Figure 3.5), and is able to obtain state-of-the-art accuracy [375] with $\Delta\hat{\texttt{FPR}}$



	Baseline	Simple	Super
$\Delta\mathbf{\hat{FPR}}$	2.1 ± 1.8%	3.0 ± 1.4%	1.8 ± 1.0%
$\mathbf{\hat{FPR}}_{\mathbf{NW}}$	14.7 ± 1.3%	11.4 ± 1.0%	12.9 ± .8%
$\mathbf{\hat{FPR}}_{\mathbf{W}}$	12.6 ± 1.3%	8.4 ± 1.0%	11.1 ± .6%

Figure 3.5: Algorithm 2, LR on `COMPAS`. $B = 101$, 10 train/test splits. Table shows mean $\hat{\text{FPR}}$ ± STD for individual models (Baseline) and ensembling methods' prediction sets.

between 1.8 – 3%. Analogous results for `German Credit` indicate statistical equivalence in fairness metrics (online Appendix). These low-single-digit disparities do not cohere with much of the literature, which often reports much larger fairness violations [344, notably]. However, most work on fair classification examines individual models, selected via cross-validation with a handful of random seeds (Section 3.2). Our results suggest that selecting between a few individual models in fair binary classification experiments is unreliable. When we instead estimate expected error by ensembling, we have difficulty reproducing unfairness in practice. Variance in the underlying models in $\hat{\mu}$ seems to be the culprit. The individual models we train exhibit radically different group-specific error rates. Our strategy of shifting focus to the overall behavior of $\hat{\mu}$ provides a solution: we not only mitigate arbitrariness, we also improve accuracy *and* usually average away most underlying, individual-model unfairness.

3.6 Discussion and Related Work

In this paper, we advocate for a shift in thinking about *individual* models to the *distribution over possible models* in fair binary classification. This shift surfaces arbitrariness in underlying model decisions. We suggest a metric of *self-consistency* as a proxy for arbitrariness (Section 3.3) and an intuitive, elegantly simple extension of the classic bagging algorithm to mitigate it (Section 3.4). Our approach is tremendously effective with respect to improving $\hat{\mathtt{SC}}$, accuracy, and fairness metrics in practice (Section 3.5).

Our findings complicate accepted truths in algorithmic fairness. For example, much work posits that there is an inherent analytical trade-off between fairness and accuracy [154, 406]. Instead, our experiments complement prior work that disputes the practical relevance of this formulation [497]. We show it is in fact typically possible to achieve accuracy (via variance reduction) and close-to-fairness — and to do so without using fairness-focused interventions.

Other research also highlights the need for metrics beyond fairness and accuracy. Model multiplicity reasons about sets of models that have similar accuracy [86], but differ in underlying properties due to variance in decision rules [68, 395]. This work emphasizes developing criteria for selecting an *individual* model from that set. Instead, our work uses the *distribution over possible models* (with no normative claims about model accuracy or other criteria) to reason about arbitrariness (App C.3). Some related work considers the role of uncertainty and variance in fairness [119, 312]. Notably, Black et al. [67] concurrently investigates abstention-based ensembling, employing a strategy that (based on their choice of variance definition) ultimately does not address the

arbitrariness we describe and mitigate (Appendix C.2).

Most importantly, we take a comprehensive experimental approach missing from prior work. It is this approach that uncovers our alarming results: almost all tasks and settings demonstrate close-to or complete statistical equality in fairness metrics, after accounting for arbitrariness (§E.4). `Old Adult` (Figure 3.3a) is one of two exceptions. These results hold for larger, newer datasets like `HMDA`, which we clean and release. Altogether, our findings indicate that variance is undermining the reliability of conclusions in fair binary classification experiments. It is worth revisiting all prior experiments that depend on cross validation or few models.

The future of fairness research. While the field has put forth numerous theoretical results about (un)fairness regarding single models — impossibility of satisfying multiple metrics [318], post-processing individual models to achieve a particular metric [260] — these results seem to miss the point. By examining individual models, arbitrariness remains latent; when we account for arbitrariness in practice, most measurements of unfairness vanish. We are not suggesting that there are no reasons to be concerned with fairness of ML models. We are not challenging the idea that actual, reliable violations of standard fairness metrics should be of concern. Instead, we are suggesting that common formalisms and methods for measuring fairness can conceal a tremendous amount of arbitrariness, which should itself be an important concern when examining the social impact of automated decision-making.

CHAPTER 4

NON-DETERMINISM AND THE LAWLESSNESS OF MACHINE LEARNING CODE

The types of arbitrariness that we describe in the prior chapters can lead to outcomes that have broader social impact, when machine-learning techniques are taken up to inform decision processes in the real world. This chapter represents early work on translating the importance of these particular types of arbitrariness for a law and policy audience, with particular attention to algorithmic fairness contexts (Chapter 3). Over time, through the development of follow-on work (Chapter 3, Cooper et al. [141]), we have come to believe that this relationship is more significant than suggested in this chapter. In particular, there remains important legal-theory work to more fully address the arbitrariness induced by stochasticity. We have begun follow-on work in this area, which we hope to complete later this year.

Chapter summary: Legal literature on machine learning (ML) tends to focus on harms, and thus tends to reason about individual model outcomes and summary error rates. This focus has masked important aspects of ML that are rooted in its reliance on randomness — namely, *stochasticity* and *non-determinism*. While some recent work has begun to reason about the relationship between stochasticity and arbitrariness in legal contexts, the role of non-determinism more broadly remains unexamined. In this paper, we clarify the overlap and differences between these two concepts, and show that the effects of non-determinism, and consequently its implications for the law, become clearer from the perspective of reasoning about ML outputs as *distributions over possible outcomes*. This distributional viewpoint accounts for randomness by emphasiz-

ing the *possible* outcomes of ML. Importantly, this type of reasoning is not exclusive with current legal reasoning; it complements (and in fact can strengthen) analyses concerning individual, concrete outcomes for specific automated decisions. By illuminating the important role of non-determinism, we demonstrate that ML code falls outside of the cyberlaw frame of treating "code as law," as this frame assumes that code is deterministic. We conclude with a brief discussion of what work ML can do to constrain the potentially harm-inducing effects of non-determinism, and we indicate where the law must do work to bridge the gap between its current individual-outcome focus and the distributional approach that we recommend.

This chapter is a licensed derivative copy of work published and awarded a Long Presentation slot at *CSLAW 2022* [138]. A longer version of this work, building on results in Cooper et al. [141], was presented at *PLSC 2023* and is in development for law-review submission.

4.1 Introduction

Legal decision logic bears some resemblance with the logic of mathematical functions in that both involve procedures for mapping inputs to outputs. When adjudicating a particular case, a magistrate assembles the available evidence, which they supply as parameters to legal rules to inform decisions. Just as with mathematical functions, there can be variations in input parameters, which correspond to variations in outcomes.[1] Kolber [324] takes this functional analogy a step further, classifying the correspondences between legal inputs and outputs into "smooth" and "bumpy" types. A smooth relationship is one for which

[1]For more general background on how legal rules function despite variation in their application, we refer the reader to Fuller [218] and Tamanaha [570].

gradual changes in inputs map to gradual changes in outputs. Bumpy relationships, in contrast, exhibit discontinuities: slight variations in inputs can map to large variations in outputs.[2] Machine learning (ML) — a discipline within the mathematical tradition — unsurprisingly seems to follow a similar logic. Classification problems resemble Kolber [324]'s concept of bumpiness; varied, continuous inputs become discretized outputs. Determining loan-worthiness, for example, is bumpy because a classification model maps personal data to a binary outcome in the set {`grant_loan`, `reject_loan`}, typically based on some underlying notion of whether the individual under consideration is likely to repay or default.

This comparison between the work of law and that of ML, in which both are reasoned about as functions, is deceptively attractive. At first glance, it seems to mirror the decades-long literature in cyberlaw that has considered the law and if/then code rules[3] to be complementary modalities that regulate and mediate human experience [39, 128, 245, 359, 361, 492]. It is thus perhaps intuitive to consider stretching this analogy further: to treat the mathematical-functional similarity of the law and ML as a rationale for christening ML as the latest type of code-imbued regulator. To stretch this even further, if ML can be fashioned to design new "microdirectives" or usher in a new era of "personalized law," as some legal scholars contend [112, 195], then perhaps ML could breathe new life into the succinct cyberlaw refrain that "code is law" [361, 492]. That is, rather than using this widely-quoted shorthand to stand in for the more-precise (but still abbreviated) "code is constitutive of law" [39, p. 675], ML code could

[2]For example, it may be reasonable to contend that tort law should be smooth, with the amount of harm caused exhibiting a direct and continuous relationship with the degree of compensation owed. However, in practice, tort law is often bumpy: defendants are either liable to provide full compensation (regardless of the particular degree of contributing to harm), or they are not liable at all [324, p. 673].

[3]Either as a type of architecture [359, 361] or a modality on its own [245].

literally be used to generate law.

And yet, while it might be appealing to take these steps to connect the nascent field of ML law with its older cyberlaw sibling, upon deeper examination the comparison between ML and the law via functions does not hold up. For one, as much legal scholarship acknowledges, the mechanism by which ML translates from inputs to outputs fundamentally differs from analogous mechanisms in the law [46, 129, 130, 264, 334, 425, 426]. The law has a variety of mechanisms — rules, standards, factors tests, etc. — each accompanied with justifications for (and amendments regarding) their use, as well as a long record in jurisprudence of their application to specific cases. In contrast, ML may behave like a function, but we often do not understand how that function works. In ML systems, we can have full access to both the inputs and subsequent outputs, while having no clear understanding of *how* the mapping from one to the other occurred. In other words, unlike the law, ML functions defy explanation and reasonable justification, which in turn raises fundamental questions about the legitimacy of using ML as a decision-making tool and muddies the ability to determine accountability when these tools cause harms [146, 161].[4] In short, ML's problem with *explainability* shows how the analogy essentially and inescapably falls short; both the law and ML may behave like functions, but

[4]Clarity of explanation in legal contexts, however, is not a given. As Fuller [218] notes, "It is easy to assert that the legislator has a moral duty to make his laws clear and understandable. But this remains at best an exhortation unless we are prepared to define the degree of clarity he must attain in order to discharge his duty. The notion of subjecting clarity to quantitative measure presents obvious difficulties. We may content ourselves, of course, by saying that the legislator has at least a moral duty to try to be clear. But this only postpones the difficulty, for in some situations nothing can be more baffling than to attempt to measure how vigorously a man intended to do that which he has failed to do. ... [However,] good intentions are of little avail. ... All of this adds up to the conclusion that the inner morality of law is condemned to remain largely a morality of aspiration and not of duty. Its primary appeal must be to a sense of trusteeship and to the pride of the craftsman" [218, pp. 42-43]. It is reasonable to argue, though out of scope for this paper, that ML does not have an analogous "sense of trusteeship" on which the public can rely.

functions that are fundamentally different in kind.

This analogy falls short in another fundamental way — one that is significant enough for us to pause attempting to close the loop between cyberlaw, code-is-law scholarship and legal scholarship about ML, but has thus-far remained under-explored. Code that follows if/then logic — the type of code addressed in cyberlaw literature [39, 128, 245, 359] — is *deterministic*: it specifies behaviors to execute (the "then") when certain, specified conditions (the "if") are met. Importantly, ML code does not execute if/then rules. Instead, the ML training process is random in nature; it exhibits *stochasticity* and *non-determinism*.[5] We explore the meaning of these terms in detail later in this paper (Section 4.2). For now, it suffices to provide an intuition: deterministic code ensures that computing with the same inputs yields the same outputs; stochasticity and non-determinism, in contrast, can cause two similar training procedures to produce vastly different results in practice [141, 210, 477].

In the remainder of this chapter, we explain how stochasticity and non-determinism play a fundamental role in the behavior of ML systems. While some legal scholarship has begun to reason about the relationship between stochasticity and arbitrariness [38, 161], the role of non-determinism more generally remains unexamined. We argue that a more precise understanding of non-determinism is essential for reasoning about questions concerning the regulability, legitimacy, and accountability of ML decision-making tools.

Our first contribution is to show that the emphasis on individual errors and error rates in existing legal scholarship is concealing other important issues in

[5]Non-determinism and stochasticity are not unique to ML, but rather are features of many types of randomized programs (including programs and protocols that predate the Internet and cyberlaw). Nevertheless, the advent of ML applications in public life, and the social valences these applications carry, has brought urgency to clarifying these concepts in relation to ML.

ML that are rooted in non-determinism. While focusing on individual outcomes and error rates for specific models is important — and intuitive, given that it parallels case-based analysis in the law — it nonetheless provides a limited view of behavior of ML. We clarify the distinction between stochasticity and non-determinism more broadly construed, and show that the effects of non-determinism, and consequently its implications for the law, instead become clearer from the perspective of reasoning about ML outputs as *distributions or patterns over possible outcomes*. The key difference is that this viewpoint accounts for randomness and other types of non-determinism by providing a window into the *possible* outcomes of ML. Importantly, this type of reasoning is not exclusive with current legal reasoning; it complements (and in fact can strengthen) analyses of individual, concrete outcomes for specific automated decisions (Section 4.2).

By illuminating the important role and potential effects of non-determinism, we then demonstrate that ML code falls outside of the cyberlaw frame, which assumes deterministic code (Section 4.3). Even if this frame can be expanded to include the stochastic elements of ML, we discuss how it cannot be extended to non-deterministic elements more generally. Lastly, we conclude with a brief discussion of what work ML can do to constrain the potentially harm-inducing effects of non-determinism, and we indicate where the law must do work to bridge the gap between its current case-based analysis of ML systems and the pattern/distributional analysis that we recommend (Section 4.4).

4.2 Non-determinism and Stochasticity

Legal literature regarding the empirical performance of ML tools tends to focus on issues of accuracy [354] [87, pp. 1249-50] [98, pp. 9,12] [128, p. 1253].[6] This work typically evaluates ML in terms of individual decision outcomes in relation to the harms these outcomes cause, and uses summary error rates to draw conclusions about a particular model's accuracy. Solely focusing on the accuracy of specific inference outcomes and summary rates can conceal other important issues implicated by non-determinism, which are also important factors to consider in legal analyses of ML technology. To make this case, we first must establish definitions for non-determinism and stochasticity, as there are nuanced differences and overlap between the two terms.

Definition 11. ***Non-determinism*** *is a property of processes for which supplying the same inputs can produce different outputs.*

As a result, non-deterministic outcomes are uncertain. This is in contrast to deterministic if/then logic, for which the same inputs produce the same outputs. Stochasticity also satisfies Definition 11; however, it places additional conditions on the form that uncertainty can take.

Definition 12. ***Stochasticity*** *is a property of non-deterministic processes whose outcomes can be reasoned about using probability theory.*

In other words, the non-determinism of stochasticity specifically comes from randomization that can be understood using probability. Following

[6] Work on fairness typically focuses on accuracy, as well, by emphasizing differences in inaccuracy via error rates, and the resulting disparate impact, for protected demographic groups.

these definitions, we can think of stochastic decision-making processes as non-deterministic; however, non-deterministic decision-making processes are not necessarily stochastic, since they cannot always be reasoned about using the laws of probability.

Machine learning is grounded in probability and statistics, and thus is fundamentally stochastic in nature. In practice, however, it is also common for ML to exhibit non-determinism beyond this stochasticity. While the formal specification for an algorithm is stochastic, its implementation and execution in software and hardware can introduce non-determinism that is not stochastic. We can attempt to apply the rules of probability to reason about this behavior, but we are not guaranteed that our conclusions will be sound. A notable example of this non-determinism comes from the popular PyTorch library.[7] When prepared for execution on a computer at training time, PyTorch makes dynamic choices regarding how to run the code, which optimize for run-time speed and, in doing so, introduce non-stochastic non-determinism to the learning process.

4.2.1 Related Work: ML Stochasticity and the Law

The legal literature that discusses uncertainty and subsequent impressions of arbitrariness in ML decision-making does not reckon with this practical reality. Rather, in talking about algorithms, and more specifically their error rates or individual outcomes, this literature regards ML in stochastic terms. For example, Bambauer et al. [38] coins the term "Small Change Makes a Big Difference"

[7] We refer to PyTorch for discussion about limiting the sources of software and hardware non-determinism in ML training pipelines. At the time of writing, PyTorch offers a "deterministic mode" that, at the cost of significant run-time slowdowns that may not be feasible for all application developers, enforce determinism in software operations (where possible).

(SCMBD) to analyze the risks to due process that can come from disproportionate outcomes on similar inputs due to the stochastic nature of ML training pipelines [38, pp. 2378-2383, pp. 2396-2397]. Their discussion makes no mention of how other sources of non-determinism further expand this category of risk.

In another recent example, Creel and Hellman [161] take a formal philosophical approach to understanding what is precisely connoted by criticisms of "arbitrariness" in ML system outputs. They break down their analysis of what is arbitrary in three different respects: "unpredictable," "unconstrained," and "unreasonable" behaviors of these systems.[8] In their discussion, they claim that arbitrariness of ML systems in itself is not the problem; rather, the problem is "the systematicity of their arbitrariness" that may "irrationally [exclude] a person from a significant number of important opportunities" [161, p. 2].[9] In relation to this claim, they add "To the extent that an algorithm governs the decision, it will produce the same result when run on the same inputs. If the **algorithm** contains a degree of **randomness** within it, ... it is still **reproducible** at a higher level of abstraction" [161, pp. 3-4, emphasis added]. That is, they describe a *model* demonstrating deterministic behavior. A particular model produced from an algorithmic learning procedure is deterministic — always producing the same output given the same input (Section 4.2.2); however, as we have discussed above, the entire procedure that produces such a model is *not*

[8] In their discussion of "arbitrary" as "unconstrained," Creel and Hellman [161, p. 3-4] call algorithms "rule-based" in close proximity to discussing legal rules and standards. We do not believe that stochastic algorithms are "rule-based" in the same sense as legal rules; however, discussing this distinction is out of scope for this paper.

[9] douek [192] makes a related but different point about shifting legal understanding away from individual outcomes. She call for a shift "from an individualistic approach to a probabilistic one" [192, p. 789]. douek makes an important intervention regarding the inevitability of error in ML applications, particularly at scale, but ultimately focuses on individual model error rates and makes an argument predicated on the ability to reason about probabilities, and thus is not examining the same concepts with which we concern ourselves here.

deterministic.

Put differently, implicit in the reasoning in Creel and Hellman [161] is that the uncertainty at play in ML can be reasoned about using probability. It is probability theory that enables the systematic, "higher level of abstraction" of reasoning about the overall, expected behavior of stochastic *algorithms*, and whether those behaviors are systematically, arbitrarily unfair (according to a particular fairness criterion). However, in contrast to abstract algorithm specifications, the implemented, run-time behavior of ML *pipelines* and *systems* introduces non-stochastic non-determinism — non-determinism that is *not* systematic, in the sense that it cannot be reasoned about analytically with the guarantees of probability theory. This is not a distinction without import; in contrast to Creel and Hellman's claim about reproducibility in relation to what we understand as stochastic-related arbitrariness, this kind of non-determinism is a well-known contributor to the reproducibility crisis in ML [80, 484]. Non-stochastic non-determinism thus suggests a different kind of arbitrariness from that discussed in Creel and Hellman [161], and it, too, can have significant impacts on normative concerns like fairness [477] (Section 4.2.3).

In short, though prior legal literature on ML and arbitrariness sometimes engages with elements of stochasticity, it does not account for the role of other forms of non-determinism. In the remainder of this section, we explain via simple synthetic examples how the presence of non-determinism calls into question essential assumptions about the fundamental nature of accuracy in ML. Moving away from analyses of individual outcomes to thinking about *distributions/patterns over possible outcomes* can expand legal scholars' understanding of the be-

havior of ML tools. In particular, reasoning about *probability* distributions over possible outcomes is useful for understanding the impacts of stochasticity (Section 4.2.2). While probability is not similarly useful for analytically reasoning about other sources of non-determinism, our approach can still highlight empirically the importance of the role of non-determinism in ML and the potential harms it can cause (Section 4.2.3).

4.2.2 Distributions over Individual Outcomes

We first consider a synthetic ML system that aims to determine individuals' creditworthiness by predicting their credit scores. The developers write a snippet of code to achieve this task — a procedure for training models to predict individuals' credit scores. The execution of this code to actually train a model exhibits stochasticity: running this one piece of code multiple times on different subsets of the training data will result in multiple trained models that vary in comparison to one another. If we were to take many such models and supply them with the same individual as input, the corresponding outputs would yield a distribution over possible credit score outcomes for that individual. We illustrate this in Figure 4.1 for two individuals. In other words, since this process yields a distribution over possible credit scores for each individual — and not just a single credit score — predicting an individual's credit score is not a deterministic function of the code written by the engineer to train ML models. Rather, credit score for an individual is a function of the procedure that this code can execute; it is a function of executing model training, which exhibits stochasticity (as a function of the specific training data examples used for training) and thus a distribution of possible outcomes for different individuals.

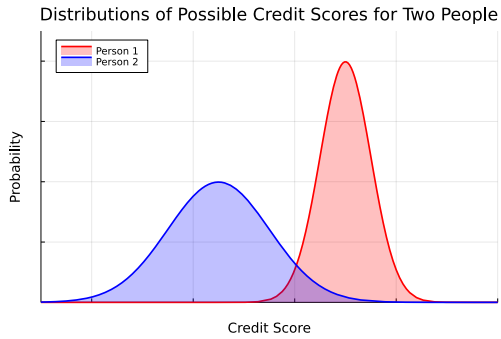


Figure 4.1: Synthetic probability distributions for possible predicted credit scores of two different individuals.

The viewpoint of distributions over possible outcomes shown in Figure 4.1 illustrates a problem: the two individuals have overlapping credit score distributions (shown in purple). This means that it is possible that there is some subset of models, produced by the stochastic training process, for which we cannot distinguish between these two individuals in terms of their credit scores. And yet, in looking at each of their distributions overall, there are all clearly cases where they do not overlap and are thus clearly distinguishable. That is, from its distributional perspective, this figure shows that it is possible to produce models that suggest contradictory results: some models are able to distinguish these individuals via different credit scores, while it is possible that some models are *not* able to discern a difference. Instead of all models from this training process having the ability to clearly distinguish between or to equate these two individuals via credit scores, both contradictory possibilities are suggested by this distributional viewpoint.

This ambiguity complicates what accuracy means for a model produced by this training process, because it is not clear what a "correct" model should do with respect to how it views these two individuals. Is it "correct" to model them as distinguishable, or "correct" to model them as indistinguishable, in terms of

their credit scores? It is impossible to say with 100% certainty, since there is no notion of ground truth credit score.[10] Put differently, this figure indicates that there is a meta-problem of not being able to draw a firm line between correctness and incorrectness for models trained by this process. This issue of being unable to draw a clear boundary between correctness and incorrectness illustrates how model output decisions can exhibit non-determinism: for the different inputs, depending on the model, the outputs for those inputs may be distinguishable or may be indistinguishable.

So far, we have limited our discussion of non-determinism to stochasticity (Definition 12) — in particular, the stochasticity resulting from training models on different subsets of the training data or from different examples drawn from the same data distribution. In practice, the other sources of non-determinism that we describe above can contribute to the results described in Figure 4.1. Moreover, it may not be immediately clear how each source contributes to the outcome predictions and impacts their associated probabilities. In other words, the distributional approach in Figure 4.1 clarifies that the predictions can fluctuate, but it conceals how stochasticity and other sources of non-determinism interact to produce those fluctuations — a point to which we return in Section 4.3, where we discuss the regulability of ML code.

For now, we observe that the legal literature discussed in Section 4.2.1 touches on the stochasticity that contributes to examples like this one, but it does so in a manner different from the distributional picture we show here. Bambauer et al. [38] discusses how stochasticity can cause *a particular model* to exhibit SCMBDs that affect due process. Similarly, Creel and Hellman [161] dis-

[10]This is in contrast to applications for which we can reasonably say that there is a ground truth, such as a computer vision system that distinguishes between cats and dogs; an example input is either a cat or a dog, not both.

cuss how *a particular model* exhibits deterministic outputs; their concern is that, at the scale of multiple decisions across multiple models for different tasks, there may be a pattern of arbitrary discrimination against certain individuals. In relation to Figure 4.1, these works engage with stochasticity at the point in which there is one model producing a concrete credit score for each individual, rather than the distribution of possible model outputs for these individuals. It is only in the setting they rely on — after we have selected a particular model to use for predicting credit scores — that we can think about deterministic outputs. That is, by picking a particular model that encodes a specific function, we have locked in a deterministic score for each individual. We can then move from reasoning about distributions over possible outcomes of credit scores for individuals, as indicated in Figure 4.1, to thinking about deterministic, concrete outcomes, which are conditional on the model we have chosen.

Given one specific model, with deterministic outcomes for each individual input, it becomes possible to perform analyses concerning the inaccuracy of individual outcomes, associated harms, and metrics like error rates to capture summary information about a model's overall performance across a sample of inputs, as Bambauer et al. [38] and Creel and Hellman [161] both do. But, importantly, at the distributional level conveyed in Figure 4.1, concepts like accuracy remain slippery. In reasoning about possible rather than specific model outcomes, this level of abstraction makes the potential areas of uncertainty in trained models — whether due to stochasticity or other sources of non-determinism — more transparent. It clarifies how the possibility of different outcomes can have the effect of muddling the distinction between correctness and incorrectness, and opens up the possibility of trying to untangle sources of non-determinism and their respective normative considerations regarding arbi-

trariness, which we discuss further in Section 4.3.

4.2.3 Patterns over Models

Reasoning over distributions of outcomes does not just apply to thinking about how outcomes for fixed individual inputs may vary based on choice of model. This view can also help reason about how non-determinism affects models trained from the same stochastic training process. Figure 4.2 shows patterns[11] over model outcomes for two models trained using the exact same procedure and, unlike the prior example, the models are trained using the same software random seed, which functions to supply the algorithm with the exact same training data. With this setup, we have controlled for every possible source of stochastic non-determinism in the training process. By using the same random seed, we should be able to consistently reproduce the same deterministic model, aligning with Creel and Hellman's conception of the training process (Section 4.2.1), and thus the two curves in Figure 4.2 should completely overlap.

The reason they do not overlap is because of non-stochastic non-determinism that affects their respective training processes differently. For example, differences in hardware random seeds, which we cannot control in software code, cause the training process to produce different models that reflect different underlying deterministic functions. Ideally, even with non-determinism in ML software packages and across hardware, repeated runs of

[11] In the camera-ready version of this paper, we used the word "distributions" to describe this effect as well, since we could not think of a better term to use at the time. However, this was a poor choice on our part, since the type of non-determinism we describe in this section cannot be reasoned about with probability, and "distribution" most typically implies that we are talking about a "probability distribution." Joan Feigenbaum suggested we use the word "pattern" instead, so we make that change here.

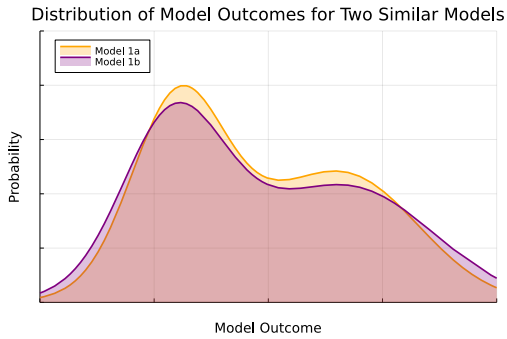


Figure 4.2: Synthetic patterns of model outcomes for two models trained on the same training data for the same task, using the same algorithm and data, but possibly different computers with different hardware random seeds. Nondeterminism in the training process yields different patterns of model outcomes.

similar or identical training processes would result in outcome distributions that are reasonably similar to each other (as, one could argue, is the case in Figure 4.2, since the curves roughly overlap). If the models' patterns of outcomes do not vary too much, then at least we can be confident (however informally) that picking any of them as the specific model to deploy is a reasonable choice, as each model indicates performance roughly representative of all the models that were trained. In other words, it might be fine to avoid the issue of drawing a line between which models are correct and which are incorrect, because all of the models are effectively the same.

Of course, though, the models are not *exactly* the same, which may have significant consequences at the more granular level of individual outcomes. They may differ in a way that is not semantically meaningful, or may exhibit more uncertainty in connection with a protected attribute value, such as a particular gender or race. Importantly, this type of uncertainty, not being stochastic in nature, is not amenable to reasoning with the tools of probability; it is not amenable to the same reasoning about arbitrariness and individual outcomes in Creel and Hellman [161], which is implied to be predicated on uncertainty due

to stochasticity.

More generally, the fact that models are not completely identical requires us to shift our thinking about ML. This pattern-level view clarifies that we should be thinking of one run of an ML training process as learning *a* possible pattern over possible outcomes, rather than *the* singularly correct pattern over possible outcomes. Additionally, unlike the synthetic example in Figure 4.2, in practice it is common for model outcome patterns to vary more significantly due to non-determinism [80, 145, 210, 477, 484, 543]. In such cases, it will not necessarily be clear if there is a representative model in the group — if there is a model that is more "correct" than the others. Once again, due to non-determinism, drawing a firm boundary between correctness and incorrectness is ill-defined. As with the example in Figure 4.1, this example similarly raises questions of how to legitimately pick a model that we can be confident will yield robust and reliable performance.[12]

This question is not just of theoretical relevance. In practice, non-determinism can cause resulting model outcome distributions to vary so much that, for a particular input, models can yield wildly inconsistent results. To make this more concrete, we describe an example in the ML literature that demonstrates the effects of such non-determinism. In recent work, Forde et al. [210] and Qian et al. [477] investigated how the impact of stochasticity[13] and non-determinism on training models using similar training procedures can impact model fairness. Qian et al. [477] published an extensive empirical study, in

[12]We could also extend the first example be plotted at the model level, rather than individual level, with outcomes on the x-axis and then the probability on the y-axis; in this case, where we only look at stochasticity as a function of the training dataset, we would have a picture that looks perhaps a lot like Figure 4.2, but the terminology would change to reflect that we could reason about this plot as containing probability distributions.

[13]And we performed the largest such study on stochasticity in fairness contexts in other work that followed [141].

which they repeatedly trained models with identical training procedures, using the same software random seed and thus exactly the same training data examples across runs. In theory, this setup should control for stochasticity in different model outputs; by using the same training data and same training procedure, the models produced should be the same. However, the realities of running ML code in practice differ from what we expect in theory. Qian et al. [477] makes the stakes of this point unimpeachably clear by comparing fluctuations in the resulting model outcome distributions. In particular, they computed common algorithmic fairness metrics to probe how fairness measurements varied for these (theoretically identical) models, and found that fairness measurements could vary by up to 12.6%. This degree of variance was so significant that, in some cases, it was possible for one trained model to pass US legal compliance rules regarding fairness thresholds on the test set, while another model could violate those same requirements [477, p. 2].

In other words, Qian et al. [477] illustrates clearly how non-stochastic nondeterminism can have a significant impact on fairness in the distribution of possible modeling outcomes. This result indicates that picking any one specific model to deploy — which then could be examined in terms of individual errors and error rates, fairness-related or otherwise — is a non-trivial task. Nondeterminism necessarily has an unpredictable role in the specific outcomes of training models, as evidenced by the resulting evaluation of test error to understand generalization. When this unpredictability leads to wide variability in metrics like fairness, this then raises fundamental questions not just about the fairness of particular models, but about the fairness of the process by which those models were trained. We may try to the best of our ability to control for models to be trained in the same way, and yet they may still exhibit vastly differ-

ent fairness levels. Given this non-determinism, how can we be sure, especially when training just a few models under limited computational resource budgets, that the model we have selected to deploy in practice is representative of what is (at least close to) maximally possible in terms of fairness?

Questions like these, let alone their answers, are not clear from looking at individual outcomes or error rates for single models alone. Instead, it is looking at patterns and distributions over outcomes that raises questions about the legitimacy model-producing processes, through indicating how the resulting models from those processes can fluctuate in important ways. This distributional/pattern-level view provides information that can help us interrogate whether the process for training ML models for a specific task is robust enough to justify the use of *any* such model produced from that process. By robust we mean that, even in the presence of non-determinism, the resulting variation in the behavior of possible ML models — whether variation in model outcome distributions, or variation in outcomes across models for particular individual inputs — is not the product of happenstance, for example resulting from a particular hardware-software interface implementation.

The example of Qian et al. [477] arguably does not meet this definition of robustness, given the large variance in fairness metrics across the distribution of models they produced.[14] This becomes especially clear when one considers how such variance in fairness could impact due process [354] [87, pp. 1249-50] [98, pp. 9,12] [128, p. 1253] — if a particular chosen model by chance demonstrates poor performance with respect to fairness, in turn leading to a greater number of unfair individual outcomes in practice.

[14]Neither do the individual models trained in Cooper et al. [141]; however, the ensemble models trained in that work are more robust in this sense.

4.3 Non-deterministic Code Is Lawless

In moving from looking at individual errors and model error rates to reasoning about distributions and patterns of outcomes, we have seen how the non-determinism inherent in ML can raise key questions concerning the legitimacy of using ML-driven processes in decision-making. We have seen, too, how non-determinism can directly effect harms at the individual level, in cases in which a training process is not sufficiently robust to guarantee that its resulting models behave similarly for key metrics, such as fairness. In short, our discussion thus-far has indicated that non-determinism can have significant, detrimental effects on the behavior of ML code. While there are different types of non-determinism, we have shown that prior work in legal ML focuses on non-determinism that is stochastic (Definition 12).[15] While this type of non-determinism is amenable to analysis using probability, other types of non-determinism in ML, such as the specifics of the hardware platform used to execute training processes, do not follow the same logic (Definition 11). As a result, work that has engaged with arbitrariness of ML decisions purely in stochastic terms has missed this crucial aspect of non-determinism and its relationship to arbitrariness.

One of the important consequences of this omission has to do with an implied, uncomfortable relationship between arbitrariness and necessity in ML. As we briefly discussed in the introduction, the stochasticity of ML is one of its core strengths that separates it from non-stochastic decision systems; it is the property that makes it possible for ML to model phenomena that are too complex to specify exhaustively using if/then deterministic rules [428]. Yet, stochasticity can also produce variable outcomes for the same inputs, and these

[15] However, upon further reflection, we realize that this work has not studied this sufficiently; we defer additional study to future work.

variations can suggest contradictions that call the reliability of ML into question (Section 4.2.2). Moreover, these potential contradictions are less intuitive to grasp than the outputs of deterministic decision processes. At times, they might even seem like software bugs, rather than an artifact of a necessary feature of ML,[16] which itself can further cast doubt on reliability. Due to this seeming double bind, it makes sense that legal literature about ML has tried to parse the cases in which the stochasticity-induced arbitrariness present concerns for the law.

However, other sources of non-determinism do not exhibit the same conflict. Lack of expressivity in hardware-software interfaces, inability to control hardware random seeds, and missing APIs for fine-grained control of run-time optimization of ML code all contribute to non-stochastic non-determinism; but, they are not necessary features of ML. They are not inherent to machine learning in theory; they are a reality of its practice. As a result, this source of non-determinism suggests potential sites for future reliable ML research. Nevertheless, in the interim, ML software and hardware ecosystems inject non-determinism into training processes, which affects the patterns of overall outcomes such that they deviate non-probabilistically from what is theoretically expected. What makes this especially challenging is that, as we demonstrated in our synthetic examples (Section 4.2), it is not always immediately clear which kind of non-determinism is responsible for impacts on the resulting distribution of outcomes, which further complicates our ability to reason about outcomes using the tools of probability.

[16]For work on the elusive boundary between bugs and inherent features in ML, please refer to Cooper et al. [146]. More generally, delineating what constitutes a bug for randomized programs is a philosophical question, which has long remained unresolved in the Programming Languages research community [330, 331].

More generally, taken together, both sources of non-determinism can make it very difficult to reason about the difference between correctness and incorrectness in ML program behaviors, thus making accuracy a fuzzy concept that is difficult to pin down.[17] And yet, in the existing legal literature on ML, the issue of inaccuracy and accuracy, particularly at the individual model level, has been a dominant theme [46, 87, 98, 129, 192, 193, 354]. For the law to adequately contend with non-determinism, we have argued that the legal literature must shift to also consider the viewpoint of distributions/patterns over outcomes, as this viewpoint indicates how non-determinism fundamentally problematizes our understanding of accuracy.

Based on this prior discussion, we now argue that this will also require a shift in the dominant thread of cyberlaw thinking that echoes the refrain that "code is law."[18] In brief, "code as law" stands in for the idea that code does the work of law; code, like the law, is a modality for regulating and mediating human behavior [245, 359]. As Grimmelmann [245] summarizes in more detail, "code is law" captures the idea that "software itself can be effectively regulated by major social institutions, such as businesses or governments. ... If other institutions can regulate software, and software can regulate individual behavior, then software provides these institutions an effective way to shape the conduct of individuals" [245, p. 1721].[19]

[17] It is also worth noting that the approximate computing concept of the trade-off between accuracy and efficiency [143, 144], and more generally using a temporal lens to analyze outcomes [567], further complicates our understanding of accuracy in ML.

[18] This phrase, which originated from work in Reidenberg [492], has been further developed and revised [245, 359, 360], and then ultimately itself codified in Lessig [361]. It has since been partially adapted to account for the new kinds of experiences that ML (particularly robotics) will mediate [34, 97].

[19] Importantly, this understanding of "code as law" grew out of legal scholarship that was reckoning with the advent of the Internet. In particular, this scholarship was concerned with "decisions about the technical future of the Internet," which it considered to be "important questions of social policy ... [that would] have the force of law even as they def[ied] many of our assumptions about law" [245, p. 1721].

In the extensive literature that has followed from Lessig [361]'s codification of the concept, various scholars have built on and problematized different aspects of "code is law" [39, 97, 245], such that it has ultimately remained a resonant and powerful frame for thinking about technology. However, the work that contends with this concept tends to (often implicitly) assume a deterministic view of code. It considers code to be a set of automated if/then rules that ensure consistent decisions — and can be institutionally regulated to ensure consistent decisions — as it works to enable and constrain human behavior [245, pp. 1721, 1728-1732] [39, p. 676] [128, p. 1253]. In this view, code can concretely specify rule-like (rather than standard-like)[20] relationships between inputs and outputs that are "free from ambiguity" [245, p. 1723]. Put simply, this conception of code maps nicely to if/then rules that resemble those in the law. Yet, as we have seen throughout this paper, the assumption of deterministic code does not hold for ML: due to its statistical nature, ML code does not operate by deterministic if/then rules. Instead, due to non-determinism, it is as if both the "if" and the "then" are fuzzy; they are not specifiable in concrete terms. It is therefore natural to ask: what does non-deterministic code do to an idea of "code as law" that is predicated on determinism?

We attempt an answer in a (sort-of) proof by contradiction. We begin by assuming that "code as law" still holds for the non-deterministic code of ML. From there, then, we would need to consider what it would mean for the law to similarly exhibit non-determinism. And this is where "code is law" immediately starts to break down. In the ideal case, the law should have deterministic outcomes based on its inputs. It can exhibit variation in the relationships between inputs and outputs, but it should not be the case that there is ran-

[20]It is perhaps interesting to consider — though out of scope in this short paper — how non-deterministic ML code may more closely resemble standards than rules.

domness or arbitrariness in those relationships [324, pp. 665-666]. In practice, non-determinism can of course occur in the law. Judicial discretion is not mechanical; given similar inputs, outputs can vary across judges (or even within the same judge) [570, p. 78]. But in spite of this non-determinism, sometimes described in relation to the "indeterminacy thesis," the law remains largely predictable.[21] Contradictions in legal rules, which interfere with predictability, are classically conceived of as "miscarriages" of the law [218, pp. 38-39]. Further, as Tamanaha [570] argues, there are generally speaking few contradictions in the law, and ambiguities can be handled through "reasoned analysis."[22]

In contrast, non-determinism — particularly non-stochastic non-determinism — does not share these qualities. Stochasticity perhaps can be considered predictable, its effects reasoned about "at a higher level of abstraction" [161, pp. 3-4] using probability theory. However, from the view of patterns over possible ML outcomes, other forms of non-determinism, ironically, inject unpredictability into ML predictions, both in an intuitive sense and more formally in its resistance to statistical analysis. Additionally, as we have seen, empirical work in ML commonly demonstrates that it can result in contradictions with significant consequences [145, 210, 477].

Moreover, unlike in ML, the legal system embodies answerability. There are actors in the system who must step forward and answer for their decisions; they must provide explanations and are subject to cross-examination [87, 584]. An-

[21] As Tamanaha [570] discusses, even if there is a relatively small number of unpredictable cases, these cases are often high-impact. General predictability in terms of case numbers should not be misconstrued as a claim that unpredictable cases have low impact. Indeterminacy and unpredictability are more frequent within the Supreme Court, and there always remains the possibility that judges could exploit "latent indeterminacy" to suit personal objectives [570, pp. 90, 122-3].

[22] For the indeterminacy thesis "To have bite it must be shown that existing legal rules form a pervasive mess of contradictions, which critical theorists have not demonstrated" [570, p. 88].

swerability in part functions to remove randomness and arbitrariness from the law. In the long run, the system undergoes an ongoing process of legitimization. In other words, the law has mechanisms for recourse, which effectively can serve (however imperfectly) to root out non-determinism; unlike ML, law treats non-determinism vis-à-vis unpredictability and contradictions like a bug, not a feature.

This indicates a fundamental incompatibility for understanding ML code as law. Whereas the law can do work to avoid non-determinism, ML inherently relies on stochasticity and *de facto* relies on non-stochastic non-determinism in state-of-the-art implementations.[23] The resulting unpredictability of ML code distinguishes it from law in that it causes ML code to evade regulation. To borrow a phrase from Jack Balkin, such "code is lawless" [34, p. 52]; the unpredictability that results from non-determinism presents key problems for thinking of code as being constitutive of law.[24]

4.4 Conclusion

Non-deterministic code may itself be lawless, but this does not mean we should entirely avoid its use[25] and that we can do nothing to better regulate its deployment in practice. On the ML side, we can strive to develop tools that ob-

[23] As briefly mentioned earlier, this is a practical reality aimed at optimizing for efficiency under conditions of limited computing resources.

[24] Balkin developed this spin on the original refrain in relation to the problem of emergence and unpredictable, unintended consequences in robotic systems. We adopt it more broadly for non-deterministic code.

[25] It does, however, seem reasonable to draw the line that ML, if lawless, should not itself be used to design law (e.g., "Micro-directives [that] will provide *ex ante* behavioral prescriptions finely tailored to every possible scenario" [112, p. 2]). ML can nevertheless still be useful in the service of law, for example, by aiding in the design of tools that help lawyers be more efficient and effective in their work [168].

tain some measure of consistency — e.g., similar model outcome distributions across training runs — even in the presence of non-determinism (stochastic or otherwise). The current push for more robust ML is in fact working to develop algorithms that leverage non-determinism to learn complex decision surfaces, but also provably have bounded effects on, for example, variance in model-training outcomes. In short, ML can do work to tighten distributions/patterns, to provide theoretical limits on error (that then have to be met in practice), and to characterize rigorous trade-offs between computational resource usage for training models and how robust resulting models can be. These are rich areas of research in ML, all of which become better-appreciated when understanding ML from a distributional/pattern-level perspective.

While ML can work improve robustness, stochastic non-determinism will always remain feature, not a bug. Legal scholarship thus needs to attend to the role of distributions over outcomes in order to fully appreciate how stochasticity contributes to uncertainty in the behavior of ML systems. A s we have seen through brief examples concerning unfairness, uncertainty and non-determinism, not just individual outcomes, can themselves implicate harms. Since the law will necessarily focus on harms, its work will be to close the gap between these two essential ways of viewing ML — to ensure that the law is able to reason about distributional aspects in such a way that these aspects serve to clarify how they relate to individual outcomes. The law must find ways to bring the distributional and the individual together, such that it can successfully bring ML to account for the harms it causes.

Part II

Taming Randomness in Scalable, Reliable Sampling and Optimization Algorithms

The promise of ML capabilities will only be realized at scale. This was true before generative AI, and is crystal clear today with respect to generative-AI systems (Part III). However, scale also makes measurement challenging: to be feasible on large-scale data, in ML we often make approximations or use heuristics that sacrifice reliability; in turn, this can sacrifice correctness in outputs. In this part, we explore such tensions in uncertainty estimation and distributed optimization algorithms, and the associated implications for law and policy. This chapter reflects work that has been published at *NeurIPS* (Spotlight and poster), *AISTATS* (poster), and *ACM EAAMO* (Oral), and the *Colorado Technology Law Journal*.

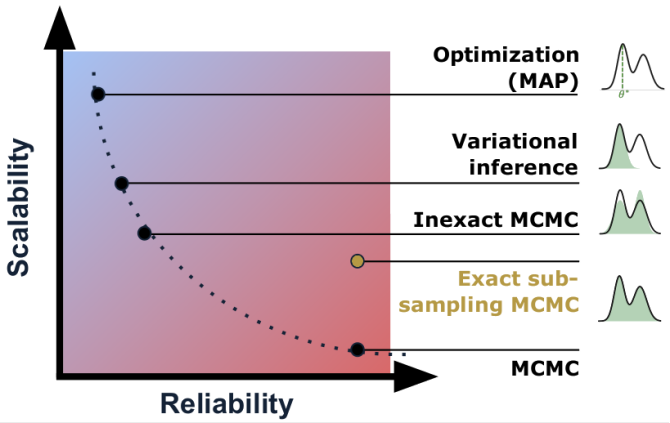


Figure 4.3: Reliability-scalability trade-off in Bayesian inference (i.e., for capturing the posterior of possible models, right). Our work (**yellow**) carefully uses subsampling to push the frontier.

First, we discuss work that examines questions related to Markov chain Monte Carlo (MCMC). MCMC is the tool of choice for reliable uncertainty estimation in Bayesian inference. Knowledge of uncertainty can help us produce more reliable models, and thus more reliable measurements of metrics. However, MCMC is also really computationally expensive, which has made it infeasible for problems of modern scale. Our work makes MCMC more efficient and scalable, while retaining reliability guarantees; it better navigates trade-offs

between these competing goals (Figure **4.3**). MCMC simulates the posterior distribution over possible-model solutions to the learning problem (in contrast to optimization's single-point estimate), which can be used to compute model-parameter uncertainties. Computing the posterior is traditionally really costly: it involves a sampling procedure that computes sums over the entire dataset at each iteration. This is part of the reason why optimization is the workhorse of modern ML. Sampling, particularly MCMC, is intractable for this setting: it typically trades-off scalability for reliability, while optimization trades-off reliability for scalability. To break out of this traditional trade-off, our work carefully introduces data subsampling to MCMC. This makes it possible to efficiently produce high-quality estimates of the posterior — to achieve *both* scalability and reliability (Figure **4.3**). Following this approach, we have developed algorithms that push the frontier of the scalability-reliability trade-off in Bayesian, in turn making reliable uncertainty estimation feasible at previously unprecedented scales (Chapter 5, Zhang et al. [642], Cooper et al. [641]).

Second, we discuss work on improved example orders for SGD-based distributed optimization (Chapter 6, Cooper et al. [140]). Such orderings represent a way that we can achieve better efficiency overall *without* sacrificing reliability. We can in fact get (in theory) guarantees of faster convergence by moving to ordering schemes that do not rely on random reshuffling every epoch. This involves a slight increase in per-iteration cost (to compute permutations), but comes with the benefit of each iteration (eventually) having an improved impact on convergence convergence. However, in practice, we need to run this ordering algorithm for a substantial number of epochs in order to see these efficiency benefits play out. Sometimes, it is not necessary to run training for this long; in turn, if we do run our method for longer, we tend to see improved

generalization.

And third, we discuss how scalability, reliablity, and trade-offs between the two exist all over ML. For example, generative-AI systems are very resource intensive: there are difficult scalability and efficiency challenges that we need to contend with in order to produce high-quality models. We argue that such trade-offs are an important abstraction for policymakers to understand the relationship between choices ML experts make and resulting ML functionality. Because such trade-offs have useful analogues in other domains that policymakers already engage with (e.g., car safety), they are legally cognizable and a natural mechanism for communication with ML experts. They can help shine a light on the barriers to accountability in stochastic algorithms (Appendix G), and help us reason rigorously about how to weaken them.

CHAPTER 5

ASYMPTOTICALLY OPTIMAL EXACT MINIBATCH METROPOLIS-HASTINGS

We begin this part with contributions to scalable, exact sampling algorithms. The work covered here [641] represents the authors' second joint collaboration on the topic of scaling reliable uncertainty estimation to new heights [642]. Several important typos are corrected from the original camera-ready paper.[1]

Chapter summary: Metropolis-Hastings (MH) is a commonly-used MCMC algorithm, but it can be intractable on large datasets due to requiring computations over the whole dataset. In this paper, we study *minibatch MH* methods, which instead use subsamples to enable scaling. We observe that most existing minibatch MH methods are inexact (i.e. they may change the target distribution), and show that this inexactness can cause arbitrarily large errors in inference. We propose a new exact minibatch MH method, *TunaMH*, which exposes a tunable trade-off between its batch size and its theoretically guaranteed convergence rate. We prove a lower bound on the batch size that any minibatch MH method *must* use to retain exactness while guaranteeing fast convergence — the first such bound for minibatch MH — and show TunaMH is asymptotically optimal in terms of the batch size. Empirically, we show TunaMH outperforms other exact minibatch MH methods on robust linear regression, truncated Gaussian mixtures, and logistic regression.

[1] In follow-on work, we also learned that obtaining the asymptotically optimal guarantees of TunaMH in practice is actually quite challenging. This is because the method changes the minibatch size every sampling iteration. With a fixed minibatch size, it is possible to pre-allocate matrix memory and just update the entries each iteration, which reduces memory overhead significantly. This is not easily attainable with a changing batch size every iteration (we spent nearly 6 months trying to do this). In practice, full-batch MH methods (especailly stochastic gradient proposal-based methods, e.g., Zhang et al. [642]), are in practice much faster in actual software.

This chapter is a licensed derivative copy of work published and awarded a spotlight at *NeurIPS 2020* [641].

5.1 Introduction

Bayesian inference is widely used for probabilistic modeling of data. Specifically, given a dataset $\mathcal{D} = \{x_i\}_{i=1}^N$ and a θ-parameterized model, it aims to compute the posterior distribution

$$\pi(\theta) \propto \exp\left(-\sum_{i=1}^{N} U_i(\theta)\right), \text{where } U_i(\theta) = -\log p(x_i|\theta) - \frac{1}{N}\log p(\theta).$$

Here $p(\theta)$ is the prior and the $p(x_i|\theta)$ give the likelihood of observing x_i given the parameter θ. We assume the data are conditionally independent given θ. The U_i have a natural interpretation as component *energy functions* with π acting as a Gibbs measure. In practice, computing $\pi(\theta)$ is often intractable and thus requires using approximate methods, such as Markov chain Monte Carlo (MCMC). MCMC uses sampling to estimate the posterior and is guaranteed to converge asymptotically to the true distribution, π [89].

The Metropolis-Hastings (MH) algorithm [263, 413] is one of the most commonly used MCMC methods. In each step, MH generates a proposal θ' from a distribution $q(\cdot|\theta)$, and accepts it with probability

$$a(\theta, \theta') = \min\left(1, \frac{\pi(\theta')q(\theta|\theta')}{\pi(\theta)q(\theta'|\theta)}\right) = \min\left(1, \exp\left(\sum_{i=1}^{N}(U_i(\theta) - U_i(\theta'))\right) \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}\right). \tag{5.1}$$

If accepted, the chain transitions to θ'; otherwise, it remains at the current state θ. This accept/reject step can be quite costly when N is large, since it entails computing a sum over the entire dataset.

Prior work has proposed many approaches to mitigate the cost of this decision step [43]. One popular approach involves introducing stochasticity: instead

of computing over the entire dataset, a subsample, or *minibatch*, is used to compute an approximation. These minibatch MH methods can be divided into two classes, *exact* and *inexact*, depending on whether or not the target distribution π is necessarily preserved. Inexact methods introduce asymptotic bias to the target distribution, trading off correctness for speedups [44, 327, 478, 479, 528]. Exact methods either require impractically strong constraints on the target distribution [390, 643], limiting their applicability in practice, or they negatively impact efficiency, counteracting the speedups that minibatching aims to provide in the first place [40, 155]. Moreover, all existing exact methods operate on the belief that there is a trade-off between batch size and convergence rate — between scalability and efficiency. Yet no prior work formally exposes this trade-off, and most prior work gives no convergence rate guarantees. Given these various considerations, it is not entirely clear how to evaluate which minibatch MH method to use.

In this paper we forge a path ahead to untangle this question. While inexact methods have been prominent recently due to their efficiency, they are not reliable: we show that the stationary distribution of any inexact method can be arbitrarily far from the target π. This means they can yield disastrously wrong inference results in practice, and it is difficult to tell just how bad those results can be.

We therefore turn our attention to exact methods and introduce *TunaMH*.[2] Compared to prior work, we make milder assumptions, which enables TunaMH to apply to a wider variety of inference tasks. More specifically, we require local rather than global bounds on the target distribution [390, 643] and do not rely

[2]TunaMH since it *tunes* the efficiency-scalability trade-off and uses a Poisson (French for "fish") variable.

on the Bernstein-von Mises approximation [43, 61, 155]. TunaMH is guaranteed to retain sample efficiency in the presence of minibatching: its convergence rate (measured by the spectral gap) is within a constant factor of standard, non-minibatch MH. More importantly, TunaMH also enables us to rigorously characterize the trade-off between scalability and efficiency. It has a hyperparameter χ, which enables tuning the trade-off between expected batch size and convergence rate.

By exposing this trade-off, our analysis raises the natural question: *is TunaMH optimal for this trade-off?* That is, could another exact algorithm use an asymptotically smaller average batch size while having the same convergence rate guarantees? We explore this in Section 5.4; under the same mild assumptions we use to derive TunaMH, we prove a lower bound on the expected batch size for *any* exact minibatch MH method that can keep a reasonable convergence rate. To our knowledge, we are the first to prove a lower bound of this nature for minibatch MH. Moreover, TunaMH is *asymptotically optimal* in balancing the expected batch size and convergence rate. It remains exact and efficient while on average using the smallest possible number of samples. In summary:

- We demonstrate that any inexact minibatch MH method can be arbitrarily inaccurate (Section 5.2.1).
- We introduce a new exact method, TunaMH (Section 5.3), with a lower bound on its convergence rate (in terms of the spectral gap) and a tunable hyperparameter to balance the trade-off between convergence rate and batch size.
- We prove a lower bound on the batch size for any exact minibatch MH method given a target convergence rate — the first such lower bound in this area. This result indicates that the expected batch size of TunaMH is asymp-

totically optimal in terms of the problem parameters (Section 5.4).

- We show empirically that TunaMH outperforms state-of-the-art exact minibatch MH methods on robust linear regression, truncated Gaussian mixture, and logistic regression (Section 5.5).

5.2 Preliminaries and Drawbacks of Prior Minibatch MH Methods

We first formally define the class of methods that we study theoretically in this paper: minibatch MH methods of the form of Algorithm 3. This class contains methods that sample a proposal from distribution q (which we always assume results in the chain being ergodic), and choose to accept or reject it by calling some randomized subroutine, `SubsMH`, which outputs 1 or 0 for "accept" or "reject," respectively.

Algorithms in this class have several notable properties. First, `SubsMH` is *stateless*: each acceptance decision is made independently, without carrying over local state associated with the MH procedure between steps. Many prior methods are stateless [44, 155, 327, 528]. We do not consider *stateful* methods, in which the decision depends on previous state; they are difficult to analyze due to running on an extended state space [25, 478]. Second, `SubsMH` takes a function that computes energy *differences* $U_i(\theta) - U_i(\theta')$ and outputs an acceptance decision. We evaluate efficiency in terms of how many times `SubsMH` calls this function, which we term the *batch size* the method uses. Third, `SubsMH` takes parameters that bound the maximum magnitude of the energy differences. Specifically, as in Cornish et al. [155], we assume:

Algorithm 3 Stateless, Energy-Difference-Based Minibatch Metropolis-Hastings

given: state space Θ, energy functions $U_1, \ldots, U_N : \Theta \to \mathbb{R}$, proposal dist. q, initial state $\theta \in \Theta$
given: parameters $c_1, \ldots, c_N$, C, M from Assumption 1, randomized algorithm `SubsMH`
loop
  sample $\theta' \sim q(\cdot|\theta)$
  define function $\Delta U : \{1, \ldots, N\} \to \mathbb{R}$, such that $\Delta U(i) = U_i(\theta) - U_i(\theta')$
  call subroutine $o \leftarrow$ `SubsMH`$(\Delta U, N, q(\theta|\theta')/q(\theta'|\theta), c_1, \ldots, c_N, C, M(\theta, \theta'))$
  if $o = 1$, **update** $\theta \leftarrow \theta'$
end loop

Assumption 1. *For some constants $c_1, \ldots, c_N \in \mathbb{R}_+$, with $\sum_i c_i = C$, and symmetric function $M : \Theta \times \Theta \to \mathbb{R}_+$, for any $\theta, \theta' \in \Theta$, the energy difference is bounded by $|U_i(\theta) - U_i(\theta')| \leq c_i M(\theta, \theta')$.*

One can derive such a bound, which can be computed in $O(1)$ time, for many common inference problems: for example, if each energy function U_i is L_i-Lipschitz continuous, then it suffices to set $c_i = L_i$ and $M(\theta, \theta') = \|\theta - \theta'\|$ (See Appendix D.10 for examples of c_i and M on common problems). Note that the `SubsMH` method may choose *not* to use these bounds in its decision. We allow this so the form of Algorithm 3 can include methods that do not require such bounds. Most existing methods can be described in this form [40, 44, 155, 327, 528]. For example, standard MH can be written by setting `SubsMH` to a subroutine that computes the acceptance rate a as in (5.1) and outputs 1 (i.e., accept) with probability a.

Such minibatch MH methods broadly come in two flavors: *inexact* and *exact*. We next establish the importance of being exact and demonstrate how TunaMH resolves drawbacks in prior work.

5.2.1 The Importance of Being Exact

Inexact methods are popular due to helping scale MH to new heights [44, 327, 478, 528]. They approximate the MH acceptance ratio to within an error tolerance (> 0), trading off exactness for efficiency gains. Surprisingly, the bias from inexactness can be arbitrarily large even when the error tolerance is small.

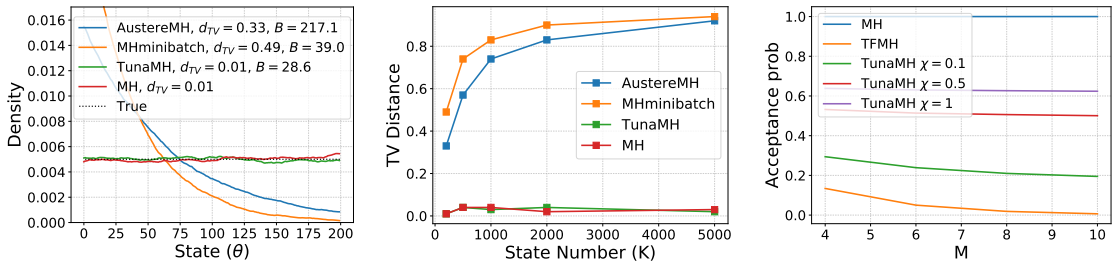
Theorem 2. *Consider any minibatch MH method of the form in Algorithm 3 that is inexact (i.e. does not necessarily have π as its stationary distribution for all π satisfying Assump. 1). For any constants $\delta \in (0, 1)$ and $\rho > 0$, there exists a target distribution π and proposal distribution q such that if we let $\tilde{\pi}$ denote a stationary distribution of the inexact minibatch MH method on this target, it satisfies*

$$\mathrm{TV}(\pi, \tilde{\pi}) \geq \delta \text{ and } \mathrm{KL}(\pi, \tilde{\pi}) \geq \rho.$$

where TV is the total variation distance and KL is the Kullback–Leibler divergence.

Theorem 2 shows that when using any inexact method, there always exists a target distribution π (factored in terms of energy functions U_i) and proposal distribution q such that it will approximate π arbitrarily poorly. This can happen even when individual errors are small; they can still accumulate a very large overall error. We prove Theorem 2 via a simple example — a random walk along a line, in which the inexact method causes the chain to step towards one direction more often than the other, even though its steps should be balanced (Appendix D.1). Note that it may be possible to avoid a large error by using some specific proposal distribution, but such a proposal is hard to know in general.

We use AustereMH [327] and MHminibatch [528] to empirically validate Theorem 2. For these inexact methods, we plot density estimates with the num-



(a) (b) (c)

Figure 5.1: Existing MH method issues. (a)-(b) Inexact methods can diverge a lot from true distribution. "d_{TV}" and "B" denote the TV distance and the batch size respectively. (c) SMH has low and TunaMH with different values of hyperparameter χ has high acceptance rates.

ber of states $K = 200$ in Figure 5.1a (see Appendix D.10.1 for using other K); the stationary distribution diverges from the target distribution significantly. Moreover, the TV distance between the density estimate and the true density increases as K increases on this random walk example (Figure 5.1b). By contrast, our exact method (Section 5.3) keeps a small TV distance on all K and estimates the density accurately with an even smaller average batch size. We also tested AustereMH on robust linear regression, a common task, to show that the error of inexact methods can be large on standard problems (Appendix D.10.1).

5.2.2 Issues with Existing Exact Methods

This observation suggests that we should be using exact methods when doing minibatch MH. However, existing approaches present additional drawbacks, which we discuss below.

Factorized MH and Scalable MH are stateless, exact minibatch methods. Factorized MH (FMH) decomposes the acceptance rate into a product of factors, which allows for rejecting a proposal based on a minibatch of data [40, 114, 125].

Truncated FMH (TFMH) is a FMH variant that maintains geometric ergodicity; it falls back on standard MH in a step when the bound on the factors reaches a certain threshold [155]. No matter how this threshold is set, we can construct tasks where TFMH is either arbitrarily inefficient (rejecting arbitrarily often, slowing convergence), or degrades entirely to standard MH.

Statement 1. *For any constant $p \in (0, 1)$, there exists a target distribution such that TFMH either has an acceptance rate which is less than p times that of standard MH, or it completely degrades to standard MH (summing over the whole dataset at each step).*

We prove this statement in Appendix D.3 using an example of a uniform distribution along a line, where we let x_i take one of two values, $\{-M/N, M/N\}$ with $M > 0$. We show that the acceptance rate of TFMH can be arbitrarily low by increasing M, which we also empirically verify in Figure 5.1c.

To improve the acceptance rate of TFMH, Scalable MH (SMH) introduces control variates, which approximate U_i with a Taylor series around the mode [155]. However, it only works with unimodal posteriors and high-quality Bernstein-von Mises approximations — conditions that do not hold for many common inference tasks.

PoissonMH is a stateless minibatch MH method adapted from an algorithm designed for scaling Gibbs sampling on factor graphs [643]. However, unlike our method, it requires strong assumptions — specifically, a global upper bound on the energy. Such an upper bound usually does not exist and, even if it does, can be very large, resulting in an impractically large batch size.

FlyMC is a stateful method, which means it uses auxiliary random variables to persist state across different MH steps [390]. It requires a lower bound on the

likelihood function, which is typically more demanding than Assumption 1 and does not have theoretical performance guarantees.

Other exact methods exist based on Piecewise Deterministic Markov Processes [61, 77]. They require regularity conditions only available for some problems, so their practical utility is limited.

5.3 TunaMH: Asymptotically Optimal Exact MH

In this section, we present our method, TunaMH, which evades the issues of prior exact methods discussed in Section 5.2.2. Like SMH [155], our method works on distributions for which an *a priori* bound on the energy differences is known (Assumption 1).

Our algorithm, presented in Algorithm 4,[3] takes as parameters $c_1, \ldots, c_N$, C, and M from Assumption 1, along with an additional hyperparameter, $\chi > 0$. It proceeds in four steps. First, like any MH method, it generates a proposal θ' from given distribution q. Second, it samples a batch size B from a Poisson distribution. This makes the expected number of energy functions U_i evaluated by our method at each step $\mathbf{E}[B] = \chi C^2 M^2(\theta, \theta') + CM(\theta, \theta')$[4]. Importantly, this means the batch size may vary from iteration to iteration,[5] and the expected size depends on θ and θ'. For example, TunaMH may tend to set B larger for larger-distance proposals with a higher $M(\theta, \theta')$. Third, it samples (with replacement)

[3]There is a typo in the camera-ready paper that persists in the Appendix (fixed here), where the proposal ratio numerator and denominator are flipped in the accept/reject step's computation of the MH ratio.

[4]Note that $\mathbf{E}[B]$ is typically $<< N$ and can be decreased using small step sizes. If, however, $\mathbf{E}[B] > N$, then we can simply use standard MH in that iteration, similar to TFMH.

[5]See the chapter summary for more details on why this causes problems in practice.

a minibatch of size B, but for each data point it samples, it has some probability of *ejecting* this point from the minibatch. Finally, it accepts the proposed θ' with some probability, computed using a sum over the post-ejection minibatch.

Our method can be derived by carefully replacing the auxiliary variables in PoissonMH with *local* Poisson variables whose distributions change each iteration depending on the pair (θ, θ') (Appendix D.4). By construction TunaMH is exact; it preserves the target distribution π as its stationary distribution. This is because TunaMH is *reversible*, meaning its transition operator T satisfies $\pi(\theta)T(\theta, \theta') = \pi(\theta')T(\theta', \theta)$ for any $\theta, \theta' \in \Theta$. This is a common condition that guarantees that a MCMC method has π as its stationary distribution [89, 365].

Compared to previous exact methods, a significant benefit of TunaMH is that we can prove theoretical guarantees on its efficiency. Specifically, its convergence speed is guaranteed to be close to standard MH and χ allows us to control how close. To show this, we lower bound the convergence rate of TunaMH in terms of the *spectral gap*, which is commonly used to characterize convergence speed in the MCMC literature [254, 365, 502, 642, 643]. The larger the spectral gap, the faster the chain converges.

Definition 13. *The* spectral gap *of a reversible Markov chain is the distance between the largest and second-largest eigenvalues of its transition operator. That is, if the eigenvalues of the transition operator are* $1 = \lambda_1 > \lambda_2 \geq \lambda_3 \cdots$*, then the spectral gap is* $\gamma = 1 - \lambda_2$.

Theorem 3. *TunaMH (Algorithm 4) is reversible with stationary distribution* π*. Let* $\bar{\gamma}$ *denote the spectral gap of TunaMH, and let* γ *denote the spectral gap of standard MH with the same target distribution and proposal distribution. Then,*

$$\bar{\gamma} \geq \exp\left(-\tfrac{1}{\chi} - 2\sqrt{\tfrac{\log 2}{\chi}}\right) \cdot \gamma.$$

Algorithm 4 TunaMH

given: initial state $\theta \in \Theta$; proposal dist. q; hyperparameter χ; Asm. 1 parameters c_i, C, M

loop

propose $\theta' \sim q(\cdot|\theta)$ and **compute** $M(\theta, \theta')$

▷ Form minibatch $\mathcal{I}$

sample $B \sim \text{Poisson}\left(\chi C^2 M^2(\theta, \theta') + CM(\theta, \theta')\right)$

initialize minibatch indices $\mathcal{I} \leftarrow \emptyset$ (an initially empty multiset)

for $b \in \{1, \ldots, B\}$ **do**

sample i_b such that $\mathbf{P}(i_b = i) = c_i/C$, for $i = 1 \ldots N$

with probability $\frac{\chi c_{i_b} CM^2(\theta,\theta') + \frac{1}{2}(U_{i_b}(\theta') - U_{i_b}(\theta) + c_{i_b} M(\theta,\theta'))}{\chi c_{i_b} CM^2(\theta,\theta') + c_{i_b} M(\theta,\theta')}$ **add** i_b to $\mathcal{I}$

end for

▷ Accept/reject step based on minibatch $\mathcal{I}$

compute MH ratio $r \leftarrow \exp\left(2 \sum_{i \in \mathcal{I}} \text{artanh}\left(\frac{U_i(\theta) - U_i(\theta')}{c_i M(\theta,\theta')(1 + 2\chi CM(\theta,\theta'))}\right)\right) \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}$

with probability $\min(1, r)$, set $\theta \leftarrow \theta'$

end loop

Intuitively, this theorem (proof in Appendix D.5) suggests the convergence rate of TunaMH is at most a constant slower than that of standard MH, and can be increased by adjusting the hyperparameter χ. Recall that χ also controls the batch size of TunaMH. Effectively, this means χ is a *dial* that allows us to directly tune the trade-off between convergence rate and batch size. When χ is large, the batch size B is large and the spectral gap ratio, $\bar{\gamma}/\gamma$, is close to 1: the larger batch size is less scalable but keeps a high convergence rate. Conversely, when χ is small, the batch size is small and the spectral gap ratio is close to 0: we trade off slow-downs in convergence rate for scalability. For example, for any $0 < \kappa < 1$, to guarantee the spectral gap ratio $\bar{\gamma}/\gamma \geq \kappa$ it suffices to set (Appendix D.6)

$$\chi = \frac{4}{(1-\kappa)\log(1/\kappa)}, \text{giving an average batch size of}$$

$$\mathbf{E}[B] = \frac{4C^2M^2(\theta,\theta')}{(1-\kappa)\log(1/\kappa)} + CM(\theta,\theta'). \tag{5.2}$$

In practice, we usually want to minimize the wall-clock time to achieve a

certain estimate error, which requires tuning χ to optimally balance scalability and efficiency. We attempt to derive a theoretically optimal value of χ in Appendix D.7 by minimizing the product of the relaxation time—a measure of the number of steps needed—and the expected wall-clock time per step. Note that this product may be loose in bounding the total wall-clock time (we leave tightening this bound to future work), making the derived χ larger than necessary. In Section 5.5 we give a simple heuristic to tune χ, which works well and is generally better than the derived value.

Theorem 3 only requires the mild constraints of Assumption 1 on the target distribution, so applies in many scenarios and compares well to other exact methods. SMH further requires a Bernstein-von Mises approximation to have guarantees on its batch size and acceptance rate. PoissonMH provides convergence rate guarantees, but demands the strong assumption that the target distribution has a global upper bound on the energy. FlyMC does not have any theoretical guarantees on performance.

5.4 Towards Optimal Exact Minibatch MH

In Theorem 3, we expose the trade-off between convergence rate and batch size in TunaMH. Here, we take this analysis a step further to investigate the limits of how efficient an exact minibatch MH method can be. To tackle this problem, we derive a lower bound on the batch size for any minibatch MH method that retains exactness and fast convergence. We then show that TunaMH is asymptotically optimal in terms of its dependence on the problem parameters C and M. In other words, it is not possible to outperform TunaMH in this sense with a

method in the class described by Algorithm 3.

Theorem 4. *Consider any stateless exact minibatch MH algorithm described by Algorithm 3, any state space Θ (with $|\Theta| \geq 2$), any $C > 0$, and any function $M : \Theta \times \Theta \to \mathbb{R}^+$. Suppose that the algorithm guarantees that, for some constant $\kappa \in (0, 1)$, for any distribution, the ratio between the spectral gap of minibatch MH $\hat{\gamma}$ and the spectral gap of standard MH γ is bounded by $\hat{\gamma} \geq \kappa\gamma$. Then there must exist a distribution π over Θ and proposal q such that the batch size B of that algorithm, when deciding whether to accept any transition $\theta \to \theta'$, is bounded from below by*

$$\mathbf{E}[B] \geq \zeta \cdot \kappa \cdot \left(C^2 M^2(\theta, \theta') + CM(\theta, \theta')\right) \tag{5.3}$$

for some constant $\zeta > 0$ independent of algorithm and problem parameters.

To prove this theorem, we construct a random walk example over two states, then consider the smallest batch size a method requires to distinguish between two different stationary distributions (Appendix D.8). The impact of Theorem 4 is three-fold:

First, it provides an upper bound on the performance of algorithms of Algorithm 3's form: in each iteration, the average batch size of any exact minibatch MH method of the form of Algorithm 3 must be set as in (5.3) in order to maintain a reasonable convergence rate. To the best of our knowledge, this is the first theorem that rigorously proves a ceiling for the possible performance of minibatch MH.

Second, TunaMH achieves this upper bound. In fact, Theorem 4 suggests that TunaMH is *asymptotically optimal* in terms of the problem parameters, C and M. To see this, observe that when we ignore κ, both expressions that bound $\mathbf{E}[B]$ in (5.2) and (5.3) are $\Theta(C^2 M^2(\theta, \theta') + CM(\theta, \theta'))$.

Thus TunaMH reaches the lower bound, achieving asymptotic optimality in terms of C and M. (Of course, this sense of "optimality" does not rule out potential constant-factor improvements over TunaMH or improvements that depend on κ.)

Lastly, this result suggests directions for developing new exact minibatch MH algorithms: to be significantly faster than TunaMH, we either need to introduce additional assumptions to the problem or to develop new stateful algorithms.

In prior work, when assuming a very concentrated posterior, some methods' batch size can scale in $O(1)$ [43, 61, 155] or $O(1/\sqrt{N})$ [155] in terms of the dataset size N while maintaining efficiency. Theorem 4 is compatible with these results, further demonstrating this is essentially the *best* dependency on N an exact minibatch MH method can achieve. We show this by explicitly assuming the dependency of C and M on N, as in SMH [155], yielding the following corollary (proof in Appendix D.9):

Corollary 1. *Suppose that C increases linearly with N ($C = \Theta(N)$) and $M(\theta, \theta')$ scales in $\Theta(N^{-(h+1)/2})$ for some constant $h > 0$. Then the lower bound in Theorem 4 becomes $\Theta(N^{(1-h)/2})$. In particular, it is $\Theta(1)$ when $h = 1$, and $\Theta(1/\sqrt{N})$ when $h = 2$.*

That is, TunaMH matches the state-of-the-art's dependency on N, and this dependency is optimal. Similarly, since C and M are the only problem parameters in the lower bound in Theorem 4, we can also get the optimal dependency on the other problem parameters by explicitly assuming the relation of them with C and M.

5.5 Experiments

We compare TunaMH to MH, TFMH, SMH (i.e. TFMH with MAP control variates) and FlyMC. We only include PoissonMH in the Gaussian mixture experiment, as it is not applicable in the other tasks. All of these methods are unbiased, so they have the same stationary distribution. To ensure fair wall-clock time comparisons, we coded each method in Julia; our implementations are at least as fast as, if not faster than, prior implementations. For each trial, we use Gaussian random walk proposals. We tune the proposal stepsize separately for each method to reach a target acceptance rate, and report averaged results and standard error from the mean over three runs. We set χ to be roughly the largest value that keeps $\chi C^2 M^2(\theta, \theta') < 1$ in most steps; we keep χ as high as possible while the average batch size is around its lower bound $CM(\theta, \theta')$. We found this strategy works well in practice. We released the code at `https://github.com/ruqizhang/tunamh`.

5.5.1 Robust Linear Regression

We first test TunaMH on robust linear regression [155, 390]. We use a Student's t-distribution with degree of freedom $v = 4$ and set data dimension $d = 100$ (Appendix D.10). We tune each method separately to a 0.25 target acceptance rate. To measure efficiency, we record effective sample size (ESS) per second—a common MCMC metric for quantifying the number of effectively independent samples a method can draw from the posterior each second [89]. Figure 5.2a shows TunaMH is the most efficient for all dataset sizes N; it has the largest ESS/second. For minibatch MH methods, Figure 5.2b compares the average

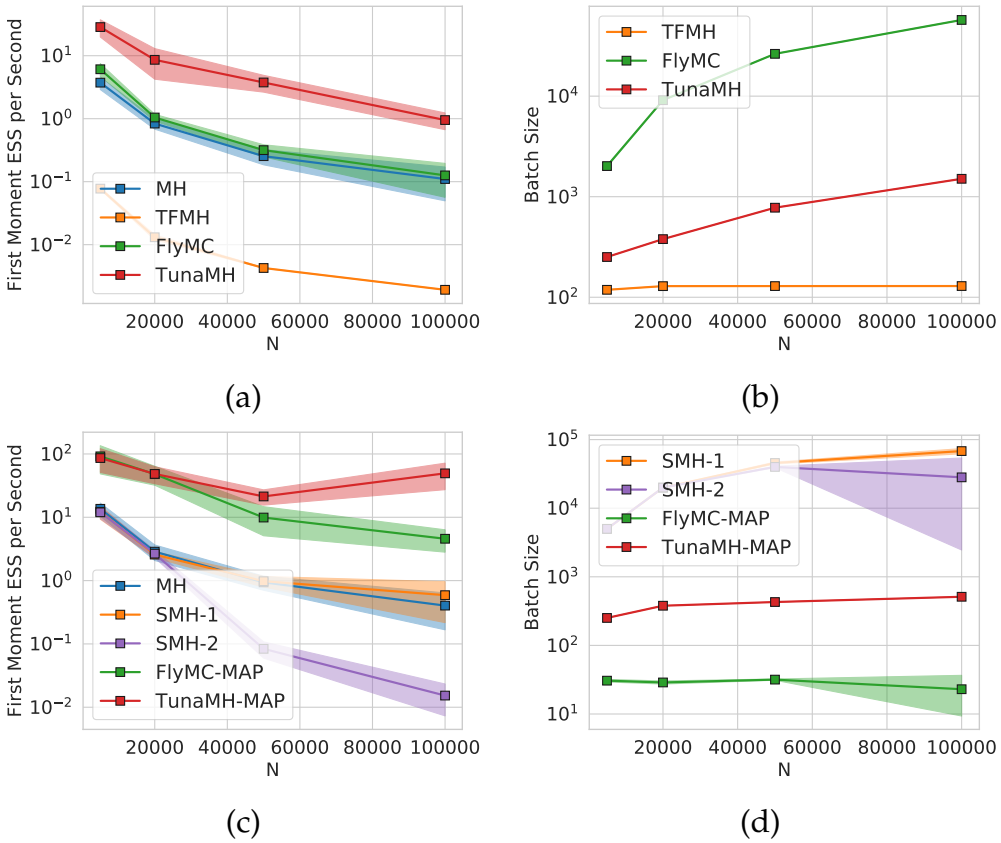


Figure 5.2: Robust linear regression, $d = 100$. (a) ESS/second without MAP. (b) Average batch size without MAP. (c) ESS/second with MAP. (d) Average batch size with MAP.

batch size. TunaMH's batch size is significantly smaller than FlyMC's — about 35x with $N = 10^5$. TFMH has the smallest batch size, but this is because it uses a very small step size to reach the target acceptance rate (Table D.1 in Appendix D.10.2). This leads to poor efficiency, which we can observe in its low ESS/second.

MAP variants Since TFMH and FlyMC have variants that use the *maximum a posteriori* (MAP) solution to boost performance, we also test TunaMH in this scheme. SMH uses MAP to construct control variates for TFMH to improve low acceptance rates. We consider both first- and second-order approximations (SMH-1 and SMH-2). FlyMC uses MAP to tighten the lower bound (FlyMC-

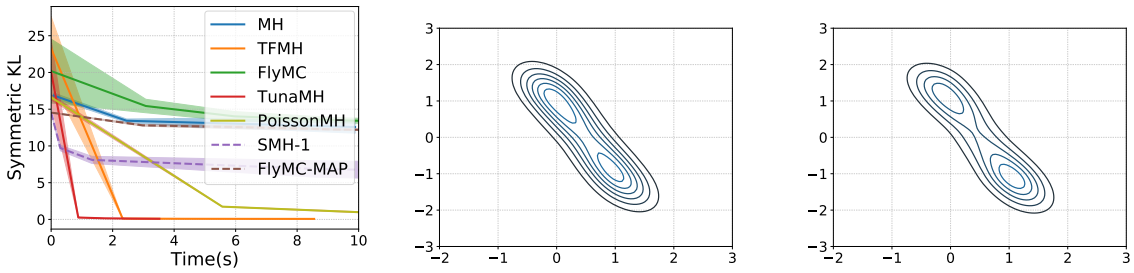
MAP). For our method (TunaMH-MAP) and MH (MH-MAP), we simply initialize the chain with the MAP solution. Figure 5.2c shows that TunaMH performs the best even when previous methods make use of MAP. With control variates, SMH does increase the acceptance rate of TFMH, but this comes at the cost of a drastically increased batch size (Figure 5.2d) which we conjecture is due to the control variates scaling poorly in high dimensions ($d = 100$).[6] FlyMC-MAP tightens the bounds, entailing a decrease in the batch size. However, as clear in the difference in ESS/second, it is still less efficient than TunaMH due to its strong dependence between auxiliary variables and the model parameters — an issue that previous work also documents [478].

5.5.2 Truncated Gaussian Mixture

Next we test on a task with a multimodal posterior, a very common problem in machine learning. This demonstrates the advantage of TunaMH not relying on MAP, because MAP is a single solution and therefore is unable to reflect all possible modes in multimodal distributions. As a result, methods that rely on MAP tuning or MAP-based control variates are unable to perform well on such problems.

We consider a Gaussian mixture. To get bounds on TunaMH, TFMH, SMH, and FlyMC, we truncate the posterior, bounding $\theta_1, \theta_2 \in [-3, 3]$ similar to Zhang and De Sa [643]. We can include PoissonMH because its required bound exists after truncation. As in Seita et al. [528], we use a tempered posterior $\pi(\theta) \propto \exp\left(-\beta \sum_i U_i(\theta)\right)$ with $N = 10^6$ and $\beta = 10^{-4}$. Figure 5.3a compares perfor-

[6] Control variates worked well in the SMH paper [155] because all experiments had small dimension ($d = 10$).



(a) (b) (c)

Figure 5.3: Truncated Gaussian mixture. (a) Symmetric KL comparison. (b) True distribution. (c) Denstity estimate of TunaMH after 1 second.

mance, showing symmetric KL versus wall-clock time. TunaMH is the fastest, converging after 1 second, whereas the others take much longer. As expected, SMH-1 performs worse than TFMH, verifying the control variate is unhelpful for multimodal distributions. FlyMC and FlyMC-MAP are also inefficient; their performance is on par with standard MH, indicating negligible benefits from minibatching.

TunaMH also performs significantly better in terms of batch size, especially in comparison to PoissonMH (Table 5.1). This is due to TunaMH's local bound on the energy, as opposed to PoissonMH's global bound. This also allows TunaMH to run on more problem types, such as robust linear (Section 5.5.1) and logistic (Section 5.5.3) regression. To illustrate the estimate quality, we also visualize the density estimate after 1 second; TunaMH's estimate (Figure 5.3c) is very close to the true distribution (Figure 5.3b), while the other methods do not provide on-par estimates within the same time budget (Appendix D.10.3).

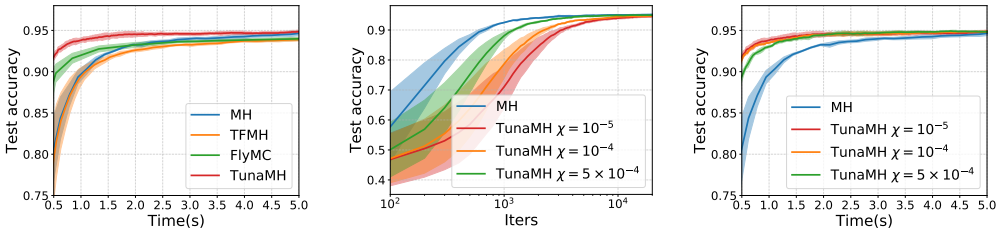
5.5.3 Logistic Regression on MNIST

Lastly we apply TunaMH to logistic regression on the MNIST image dataset of handwritten number digits. Mirroring the work of FlyMC [390], we aim to classify 7s and 9s using the first 50 principal components as features. We set $\chi = 10^{-5}$ following our heuristic. In Figure 5.4a we see that TunaMH is the fastest of all methods to converge, as measured by wall-clock time. We also compare average batch size in Table 5.1. TunaMH's average batch size is 4× smaller than FlyMC's. TFMH again has the smallest batch size, but sacrifices efficiency by using a small step size in order to achieve the target acceptance rate. Thus, overall, TFMH is again inefficient in these experiments.

Table 5.1: Avg. batch size ± SE from the mean on 3 runs. PoissonMH not applicable to logistic reg.

Tasks	TFMH	FlyMC	PoissonMH	TunaMH
Gaussian Mixture	13.91 ± 0.016	811.52 ± 234.16	3969.67 ± 327.26	86.45 ± 0.04
Logistic Regression	39.28 ± 0.12	1960.19 ± 150.96	—	504.07 ± 0.33

Effect of Hyperparameter χ To understand the effect of χ in TunaMH, we report results with varying χ. Figure 5.4b plots test accuracy as a function of the number of iterations. As χ increases, TunaMH's convergence rate approaches standard MH. This verifies our theoretical work: χ acts like a dial to control convergence rate and batch size trade-off—mapping to the efficiency-scalability trade-off. Figure 5.4c shows TunaMH's wall-clock time performance is not sensitive to χ, as the performance is superior to standard MH regardless of how we set it. However, χ needs to be tuned in order to achieve the best performance. Previous methods do not have such a dial, so they are unable to control this trade-off to improve the sampling efficiency.



(a) (b) (c)

Figure 5.4: MNIST logistic regression. (a) Test accuracy comparison. (b)-(c) TunaMH's test accuracy for various χ. Batch size for $\chi = 10^{-5}, 10^{-4}, 5 \times 10^{-4}$ is 504.07, 810.35 and 2047.91 respectively.

5.6 Conclusion and Future Work

After demonstrating that inexact methods can lead to arbitrarily incorrect inference, we focus our work in this paper on exact minibatch MH methods. We propose a new exact method, TunaMH, which lets users trade off between batch size and guaranteed convergence rate—between scalability and efficiency. We prove a lower bound on the batch size that any minibatch MH method must use to maintain exactness and convergence rate, and show TunaMH is asymptotically optimal. Our experiments validate these results, demonstrating that TunaMH outperforms state-of-the-art exact methods, particularly on high-dimensional and multimodal distributions.

To guide our analysis, we formalized a class of stateless, energy-difference-based minibatch MH methods, to which most prior methods belong. While TunaMH is asymptotically optimal for this class, future work could develop new exact methods that are better by a constant factor or on some restricted class of distributions. It would also be interesting to develop effective theoretical tools for analyzing stateful methods, since these methods could potentially bypass our lower bound.

CHAPTER 6

COORDINATING DISTRIBUTED EXAMPLE ORDERS FOR PROVABLY ACCELERATED TRAINING

We next investigate how there are better-than-random example orders for SGD that improve convergence rate in distributed settings.

Chapter summary: Recent research on online Gradient Balancing (GraB) has revealed that there exist permutation-based example orderings for SGD that are guaranteed to outperform random reshuffling (RR). Whereas RR arbitrarily permutes training examples, GraB leverages stale gradients from prior epochs to order examples — achieving a provably faster convergence rate than RR. However, GraB is limited by design: while it demonstrates an impressive ability to scale-up training on *centralized* data, it does not naturally extend to modern *distributed* ML workloads. We therefore propose *Coordinated Distributed GraB* (CD-GraB), which uses insights from prior work on kernel thinning to translate the benefits of provably faster permutation-based example ordering to distributed settings. With negligible overhead, CD-GraB exhibits a linear speedup in convergence rate over centralized GraB and outperforms distributed RR on a variety of benchmark tasks.

This chapter is a licensed derivative copy of work published at *NeurIPS 2023* [140].

6.1 Introduction

Random reshuffling, which samples training-data examples without replacement, has become the *de facto* example-ordering method in modern deep-

learning libraries [476], given that it tends to accelerate optimizer convergence in practice. However, some recent theoretical work has identified cases in which random reshuffling can lead to data orderings that have a poor effect on convergence [165, 488, 639].

This has encouraged a line of research to investigate if there exist provably better permutation-based orderings that afford greater scalability in training [384, 385, 419]. Notably, Lu et al. [384] connects permuted-order SGD to the *herding problem* [261], and proposes the herding-based online Gradient Balancing algorithm (GraB), which converges provably faster than random reshuffling, and does so with little memory or computational overhead. In fact, in follow-on work, Cha et al. [115] proves that GraB is optimal: in theory, GraB is the fastest possible permutation-based example ordering algorithm.

These results are very exciting, suggesting that GraB should unseat random reshuffling as the example ordering method-of-choice for SGD; however, they only hold with respect to a *single* machine. GraB is optimal in settings with *centralized* data, but does not naturally translate to problems of modern-ML scale, which demand that training workloads be distributed across *multiple parallel* workers that each only have access to a subset of the training data. This drawback raises an important question: *Can we simultaneously achieve the scalability benefits of distributed training and provably faster permutation-based example ordering for SGD — both in theory and in practice?*

In this chapter, we show that it is indeed possible to attain these twin objectives. To do so, we suggest the online **C**oordinated **D**istributed **Gra**diant **B**alance algorithm (CD-GraB), which leverages insights from kernel thinning to elevate the herding framework of centralized GraB (GraB) to the parallel set-

ting. Felicitously, as a side effect, this choice of formulation brings about positive practical performance benefits (that can also improve the empirical behavior of centralized GraB). Using the exact same assumptions as the original GraB paper, **we show analytically that coordinating example orders across parallel workers leads a linear speedup in convergence rate**. For T epochs and m parallel workers, each with access to n examples, CD-GraB's convergence rate is $\tilde{O}((mnT)^{-2/3})$ on smooth, non-convex objectives and $\tilde{O}((mnT)^{-2})$ under the Polyak-Łojasiewicz (P.L.) condition.[1]

We run a series of experiments to verify these improvements in practice, implementing CD-GraB on a single node that distributes computation across multiple GPUs. We also run an ablation study in order to disentangle the benefits of parallelism from the positive side effects of using kernel thinning to formulate the CD-GraB algorithm. Similar to how centralized GraB demonstrates improved generalization over centralized random reshuffling (RR), we observe that CD-GraB exhibits improved generalization over distributed random reshuffling (D-RR). Altogether, the success of our work suggests a new distributed training paradigm to explore in future work, which we call the *Order Server* (Section 6.6). In summary, we:

- Propose the online **C**oordinated **D**istributed **Gra**dient **B**alancing (CD-GraB) algorithm, which enables provably accelerated training using SGD in the parallel setting (Section 6.3);
- Prove that the convergence rate for CD-GraB exhibits a linear speedup over GraB, using the exact same assumptions as the original GraB paper (Section 6.4);

[1] In this paper, we use $\tilde{O}$ by convention to hide logarithmic factors in the problem parameters.

- Produce extensive empirical validation of CD-GraB's improved scalability on a variety of tasks in deep learning and on large-scale logistic regression (Section 6.5).

6.2 Preliminaries and Related Work

In this section, we discuss the preliminaries and prior scholarship on permutation-based example ordering, with particular attention paid to the centralized online Gradient Balancing Algorithm (GraB) [384]. This lays the groundwork for how our coordinated, distributed GraB algorithm (Section 6.3) imparts the efficiency guarantees of GraB to the parallelized regime (Section 6.4).

Ordering data examples during training. Training a model can be formulated as minimizing a differentiable loss function $f : \mathbb{R}^d \to \mathbb{R}$ over N data examples. The goal of this minimization is to obtain the target model weights $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} f(\boldsymbol{w})$, where $f(\boldsymbol{w}) = \frac{1}{N}\sum_{j=1}^{N} f(\boldsymbol{w}; j)$, for which $f(\boldsymbol{w}; j)$ denotes the loss incurred on the j-th example. A typical training process iteratively updates the model parameters $\boldsymbol{w}$ by scanning over the N data examples repeatedly, with t-th scan (or epoch) following

$$\boldsymbol{w}_t^{j+1} = \boldsymbol{w}_t^j - \alpha \nabla f(\boldsymbol{w}_t^j; \pi_t(j)), \quad \forall\, j \in [N], \tag{6.1}$$

where α denotes the learning rate, and $\pi_t : [N] \to [N]$ denotes a permutation ordering[2] adopted in the t-th epoch from which the examples are chosen to com-

[2]While without-replacement orderings are most common in large-scale learning [75], ordering strategies need not be permutations, e.g., with-replacement sampling [386, 438, 520] or curriculum learning [244, 397, 555].

pute gradients, $\boldsymbol{w}_t^1$ denotes the initial model weights for the t-th epoch, and $\boldsymbol{w}_t^j$ denotes the model weights after $j-1$ gradient updates in the t-th epoch.[3]

The choice of ordering π can have a significant effect on optimizer performance. Two popular methods, which can demonstrate convergence speedups in practice, are 1) random reshuffling (RR) [631], for which the permutations are random and differ over epochs, and 2) Shuffle Once (SO) [55, 252], for which a random permutation is computed once and remains fixed for all epochs. Recht and Ré [490] conducted the first theoretical investigation of RR, while subsequent works like Yun et al. [639] and De Sa [165] have given counterexamples in which RR leads to orderings that have a poor effect on convergence. Altogether, many studies indicate that RR and SO only provide efficiency benefits under certain conditions [251, 259, 415].

These limitations of RR and SO have motivated research to identify permutations that outperform random ones. Rajput et al. [488] introduces an RR variant that achieves improved convergence for quadratics by reversing the ordering every other epoch. Other non-RR-based methods pick efficient orderings based on correlations between adjacently selected examples. In a recent line of work, Lu et al. [385] proves that faster convergence is possible for SGD when the averages of consecutive stochastic gradients converge faster to the full gradient. Based on this result, in follow-on work Lu et al. [384] proposes the centralized online Gradient Balancing algorithm (GraB), which outperforms RR, and upon which we base this work.

[3]Note that we write (6.1) in terms of per-example-j gradients.

6.2.1 GraB: Optimal, online, permutation-based example ordering for centralized ML

GraB is a permutation-based example-ordering algorithm that identifies provably better-than-random orderings *in centralized, single-node settings* for SGD. GraB finds such orderings by leveraging information in stale stochastic gradients from previous epochs to guide ordering in the next epoch. More formally, for smooth, non-convex objectives, Lu et al. [384] proves that any permutation π^* that guarantees

$$\max_{k\in[N]} \left\| \sum_{j=1}^{k} \nabla f(\boldsymbol{w}; \pi^*(j)) - \nabla f(\boldsymbol{w}) \right\|_\infty = \tilde{O}(1)$$
$$(\nabla f(\boldsymbol{w}) \text{ is the average gradient}), \tag{6.2}$$

will yield a convergence rate of $\tilde{O}((NT)^{-2/3})$ (for epochs T) for SGD, which is superior to the $O(N^{-1/3}T^{-2/3})$ convergence rate of random reshuffling [415].

GraB's connection to herding and balancing. To find such a permutation π^*, Lu et al. [384] connect (6.2) to the *herding problem* and vector *balancing* [261, 620]. Understanding why GraB does not naturally extend to the distributed setting — and our main contributions (Sections 6.3 and 6.4) — requires some additional details on the fundamentals of herding:

Given N vectors[4] $\{z_j\}_{j=1}^{N}$ ($z_j \in \mathbb{R}^d$), $\left\|z_j\right\|_2 \leq 1$ ($\forall j$), herding identifies a permutation π^* such that

$$\max_{k\in[N]} \left\| \sum_{j=1}^{k} \left(z_{\pi^*(j)} - \bar{z} \right) \right\|_\infty = \tilde{O}(1), \qquad \text{where } \bar{z} = \tfrac{1}{N} \sum_{j=1}^{N} z_j. \tag{6.3}$$

[4]Herding does not have an optimization context. Here, N does *not* refer to the number of data examples used in training (6.1); rather, $N \in \mathbb{Z}^+$ describes the size of a set of arbitrary vectors. We slightly abuse notation because we execute the herding subroutine on exactly N gradients (Section 6.3), which happen to equal the number of N examples.

It is clear that (6.3) generalizes (6.2), which is a specific case of herding in an optimization setting.

Harvey and Samadi solve (6.3) with a method called *balancing* [261]. Balancing uses a *signed* version of the herding problem to optimize any given permutation π to reduce the bound in (6.3). That is, balancing formulates the signed herding problem

$$\max_{k\in[N]} \left\| \sum_{j=1}^{k} s_{\pi(j)} \left(z_{\pi(j)} - \bar{z} \right) \right\|_\infty, \quad \text{where } \{s_j\}_{j=1}^{N} \in \{+1, -1\}. \tag{6.4}$$

Given a group of such signs $\{s_j\}_{j=1}^{N}$ and an arbitrary permutation π, Harvey and Samadi prove that Algorithm 5 produces a new permutation π' such that

$$\max_{k\in[N]} \left\| \sum_{j=1}^{k} \left(z_{\pi'(j)} - \bar{z} \right) \right\|_\infty \leq \frac{1}{2} \max_{k\in[N]} \left\| \sum_{j=1}^{k} s_{\pi(j)} \left(z_{\pi(j)} - \bar{z} \right) \right\|_\infty + \frac{1}{2} \max_{k\in[N]} \left\| \sum_{j=1}^{k} \left(z_{\pi(j)} - \bar{z} \right) \right\|_\infty.$$

This says that, with new permutation π', the objective of (6.3) now approaches the bound of (6.4). Importantly, recent advances show that it is quite cheap to find a group of signs, such that (6.4) is on the order of $\tilde{O}(1)$ (e.g., Alweiss et al. [20], in Algorithm 6). We are therefore able to call Algorithm 5 repeatedly, which will eventually obtain the π^* that solves the $\tilde{O}(1)$ herding objective in (6.3).

GraB's application of herding to gradient balancing. Lu et al. [384] applies this framework of herding and balancing to develop GraB, i.e., to minimize (6.2). The main challenge for the success of this approach is to find the right gradients z_j in the optimization context of (6.2). Notably, the herding and balancing framework requires the vector mean $\bar{z}$ in advance. To satisfy this requirement, GraB "centers" the gradient vectors using a *stale mean*. That is, GraB runs the herding algorithm on vectors that are defined as

Algorithm 5 Reordering Vectors based on Balanced Signs [Harvey and Samadi [261]]

input: a group of signs $\{s_j\}_{j=1}^N$, initial order π
initialize: two order-sensitive lists $L_{\text{pos}} \leftarrow [\,]$, $L_{\text{neg}} \leftarrow [\,]$.
for $j = 1 \dots N$ **do**
  $L_{\text{pos}}.\text{append}(\pi(j))$ **if** s_j is $+1$ **else** $L_{\text{neg}}.\text{append}(\pi(j))$.
end for
return: new order $\pi' \coloneqq \text{concat}(L_{\text{pos}}, \text{reverse}(L_{\text{neg}}))$.

$$z_j = \nabla f(\boldsymbol{w}_t^j; \pi_t(j)) - \frac{1}{N} \sum_{p=1}^{N} \nabla f(\boldsymbol{w}_{t-1}^p; \pi_{t-1}(p)), \tag{6.5}$$

where $\boldsymbol{w}_t^p$ denotes the model weights after $p-1$ updates in the t-th epoch, and π_t denotes the permutation adopted in the t-th epoch. Lu et al. [384] proves that this definition of z_j preserves the benefits of balancing with negligible noise or overhead. The only overhead comes from storing the running average of the gradients in epoch $t-1$ to "center" the gradients in the subsequent epoch t.

With this approach, Lu et al. [384] proves that GraB demonstrates more efficient convergence than RR for SGD. Better still, Chat et al. [115] demonstrates that GraB is in fact the *optimal* permutation-based ordering method for SGD: in theory, it is not possible to produce a permutation-based ordering in the centralized setting that achieves a faster convergence rate for SGD.

Despite GraB's clear benefits over RR, it assumes local access to all examples. This assumption does not hold for popular, modern, parallel settings (e.g., parameter server [370]), in which workers only have access to subsets of examples. No present work has attempted to investigate GraB's applicability to this setting. While some work has studied distributed RR (D-RR) [285, 392, 507, 638], it remains an open question if GraB's efficiency benefits for SGD can be conferred

to the modern-scale, distributed-ML setup.

6.3 CD-GraB: A Provably Efficient Ordering Algorithm for Distributed Training

Our main contribution is to elevate GraB to the parallel regime, so that distributed training can enjoy the efficiency benefits of provably better example ordering. Based on the preliminaries, we can now explain why this is not a straightforward task: **While GraB achieves the optimal convergence rate for SGD on centralized data, it does not naturally translate to a distributed setting** (Section 6.3.1). Our key insights for resolving these problems are to reformulate the herding framework in Lu et al. [384] to work in parallel, and to leverage insights from kernel thinning [47, 182, 183] to derive the *online* PairBalance algorithm, which solves this parallelized herding objective (Section 6.3.2). Lastly, we present the full-stack CD-GraB algorithm that makes our solution work in practice (Section 6.3.3). The server implements online PairBalance, which coordinates gradient information from the distributed workers in training epoch t in order to determine a provably efficient example order for the next epoch $t + 1$ (Section 6.4).

6.3.1 Issues with GraB in the distributed setting

To clarify the issues with distributing GraB, we first need to define the distributed training setup more precisely. We consider the standard data-parallel training regime with m parallel workers, where each worker keeps a copy of the

model weights $\boldsymbol{w} \in \mathbb{R}^d$ and maintains $n = N/m$ local examples.[5] As in many data-parallel training applications,[6] such as geo-distributed model training [637], we assume *the data examples cannot be shared or moved across workers*. More formally, this setup can be expressed as

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \left[f(\boldsymbol{w}) = \frac{1}{m} \sum_{i=1}^{m} f^i(\boldsymbol{w}) \right] \quad \text{with} \quad f^i(\boldsymbol{w}) = \frac{1}{n} \sum_{j=1}^{n} f^i(\boldsymbol{w}; j), \tag{6.6}$$

where $f^i(\boldsymbol{w}; j) : \mathbb{R}^d \to \mathbb{R}$, $j \in [n]$, denotes the loss incurred on the j-th example on the i-th worker for model weights $\boldsymbol{w}$. We can now consider running (6.1) using this setup, for which each worker scans over their n local-data examples using (potentially) different permutations. We denote $\pi_{t,i} : [n] \to [n]$ as the permutation-based ordering adopted on the i-th worker in the t-th training epoch. Adjusting (6.1) to accommodate the setup in (6.6), the update to the model can be summarized as

$$\boldsymbol{w}_t^{j+1} = \boldsymbol{w}_t^j - \frac{\alpha}{m} \sum_{i=1}^{m} \nabla f^i(\boldsymbol{w}_t^j; \pi_{t,i}(j)), \quad \forall j \in [n]. \tag{6.7}$$

That is, in epoch t, each worker i selects their respective, local j-th example according to $\{\pi_{t,i}\}_{i=1}^n$ in order to compute stochastic gradients (Appendix, Chapter E).

Following this setup, Algorithm 5 no longer guarantees the $\tilde{O}(1)$ bound to the herding problem (6.3), a bound that is valid only when *all* data examples can be permuted *freely* [261]. This constraint is fine for centralized GraB, but,

[5] Without loss of generality, we assume the N examples are divided evenly among the m workers and n is even.

[6] One such popular paradigm is federated learning, in which edge devices collaboratively train a model via small local updates [400, e.g.]. Federated learning typically involves highly imbalanced loads, heterogeneous data, partial user participation, and additional privacy-preserving mechanisms. These characteristics are orthogonal to what we consider here for example order. If we were to allow for such data organization, we would need to assume non-global communication per iteration or additional constraints on how global communication occurs. For CD-GraB, we focus on the regime of using parallelism to accelerate training.

in distributed training, parallel workers only have access to a *subset* of examples. Distributed training requires that *worker-specific permutations only involve the examples in their respective local subsets*. Further, recall that GraB uses stale means to center gradients (6.5) in order to solve the herding objective. This, too, causes problems in distributed training. In practice, it is typical to employ larger learning rates α for greater scalability [549]; larger α increases the discrepancy between averaged gradients in adjacent epochs, which, in turn, would make GraB's use of stale means unreliable.

6.3.2 Our efficient solution: parallel herding & pair balancing

To address the limitations presented in the prior section, which preclude the direct application of GraB to distributed training, we will need to **1) reformulate the herding problem to fit the parallel setting, and 2) redesign how to do gradient balancing**, such that it both solves our new herding formulation and allows for reliability with higher learning rates. We now present our solution to both these problems; we introduce the *parallel herding* problem and the online PairBalance subroutine that solves it.

Parallel herding. To extend herding to the parallel setting, consider the following setup: There are m workers, which each have local access to n vectors. Let $z_{i,j} \in \mathbb{R}^d$ denote the vector indexed by j on the i-th worker. Assuming $\left\|z_{i,j}\right\|_2 \leq 1 \quad (\forall i \in [m], \forall j \in [n])$, the goal of parallel herding is to find m permutations, $\pi_1, \pi_2, \ldots, \pi_m$ where $\pi_i : [n] \to [n] \ \ (\forall i \in [m])$, so as to minimize:

$$\max_{k \in [n]} \left\| \sum_{j=1}^{k} \sum_{i=1}^{m} \left(z_{i,\pi_i(j)} - \bar{z} \right) \right\|_\infty, \quad \text{with} \quad \bar{z} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} z_{i,j}. \tag{6.8}$$

When directly comparing (6.8) with (6.3), it is clear that parallel herding differs in two notable ways from the original herding problem. First, each permutation $\pi_i : [n] \to [n] \;\; (\forall i \in [m])$ only decides the ordering of the n vectors that are associated with worker i. Second, the prefix sum taken in the objective norm is accumulated over all the workers (the inner sum from $i = 1 \dots m$). This formulation naturally captures the setting in a distributed environment: **workers need to decide permutations collaboratively, and the worker-specific vectors are processed simultaneously rather than sequentially**.

Given that this formulation fits the distributed setting, we next need to show that parallel herding does in fact address the limitations posed by centralized GraB: that it is possible recover the original $\tilde{O}(1)$ herding bound, and that we can solve the issue of unreliable stale gradients (Section 6.3.1). The solution that we present in the remainder of this section is a new vector balancing subroutine: online PairBalance. To give an intuition, as its name suggests, online PairBalance leverages insights from kernel thinning to *balance* vector differences over vector *pairs*. This also eliminates the need to perform vector centering, and thus solves the stale mean problem.

Using kernel thinning to solve parallel herding. We call our solution to the parallel herding objective (6.8) *pair balancing*, which we derive from key insights in *kernel thinning* [47, 182, 183]. In particular, Dwivedi and Mackey show that it is possible to solve the herding objective in $\tilde{O}(1)$ **by only examining differences on *pairs of examples*** [182]. They derive an algorithm that generalizes the subroutine in Algorithm 6 [20], which solves herding in $\tilde{O}(1)$ (Section 6.2), and does so by operating only on vector-pair differences.[7] This comes with a very

[7]Dwivedi and Mackey minimize the maximum mean discrepancy (MMD) between a selected coreset and an empirical distribution [182]. They develop a new self-balancing Hilbert walk on differences of *pairs of examples* to select exactly half of the dataset points, and solve

Algorithm 6 PairBalance (the server runs this online)

▷ The inputs, outputs and subroutine for this algorithm are order-sensitive

input: current running sum $\boldsymbol{r}$, paired vectors z_1, z_2

compute: $s, \boldsymbol{r} \leftarrow$ RandomizedBalance$(\boldsymbol{r}, z_1 - z_2)$
return: s (sign for z_1), $-s$ (sign for z_2), $\boldsymbol{r}$ (updated running sum)

▷ Adapted from Alweiss et al. [20]
define subroutine: RandomizedBalance$(\boldsymbol{r}, \boldsymbol{c})$
  compute: $p \leftarrow \frac{1-\langle \boldsymbol{r}, \boldsymbol{c} \rangle}{2}$
  compute: $s \leftarrow +1$ with probability p; $s \leftarrow -1$ with probability $1 - p$
  update: $\boldsymbol{r} \leftarrow \boldsymbol{r} + s\boldsymbol{c}$
  return: s, $\boldsymbol{r}$

useful property: eliminating the requirement of knowing the maximum vector norm ahead of time and centering the vectors (i.e., making all the vectors sum to zero) in order to solve the herding problem. This is the key to solving the parallel herding objective (6.8) in $\tilde{O}(1)$, and elevating the benefits of GraB to a distributed setting.

Following Dwivedi and Mackey [182], we will balance over paired vectors, and will do so in an *online* fashion (Section 6.3.3). This eliminates GraB's requirement of using a stale mean to center gradient vectors (Section 6.2.1), but still minimizes the parallel herding objective to $\tilde{O}(1)$. We defer proving this result to Section 6.4, and first describe our concrete algorithm. Online PairBalance applies Algorithm 5 on the"flattened" and "paired" sequence of all of the workers' paired-difference gradients, i.e.,

$$\boldsymbol{y}_{n(k-1)+i} = \boldsymbol{z}_{i,2k-1} - \boldsymbol{z}_{i,2k}, \quad \forall k \in [\frac{n}{2}], \quad i = 1 \dots m.$$

That is, we fit these ordered-paired differences $\{\boldsymbol{y}_i\}_{i=1}^{mn/2}$ into the herding and

coreset selection by iteratively halving the input vector sequence into balanced coresets then selecting and refining a candidate coreset to minimize MMD with the input sequence.

balancing framework (Algorithm 5): if sign s is associated with $\boldsymbol{y}_{n(k-1)+i}$, then $z_{i,2k-1}$ and $z_{i,2k}$ receive s and $-s$, respectively.

6.3.3 The full-stack CD-GraB algorithm

Having solved the parallel herding problem with pair balancing, we now demonstrate how to bring everything together in an optimization context to *coordinate distributed gradient balancing* for distributed training. That is, we can now introduce our full-stack CD-GraB algorithm, which trains models in a distributed setting (Section 6.3.1) while efficiently ordering the examples by using PairBalance (Section 6.3.2, Algorithm 6) in an online manner.

We describe CD-GraB at two levels of abstraction: a high-level illustration (Figure 6.1, steps **1–7**) and a detailed pair of worker-server algorithm statements (Figure 6.2). Since the workers only have access to a subset of the training data, in parallel they compute local, per-example stochastic gradients and send them to the server. The server simultaneously calls PairBalance online (Algorithm 6), which coordinates information from all the workers' gradients (i.e., using adjacent example-specific gradients) to determine the next epoch's worker-specific permutations. In more detail:

In epoch t, (Figure 6.1, step **1**) the two workers have permutations $\pi_{t,1}$ and $\pi_{t,2}$, respectively. Each worker computes per-example gradients $\boldsymbol{g}_j^i$ (**2**; Algorithm 7:4), and sends them to the server (**3**; Algorithm 7:5). The server we implement functions as a parameter server [370]: it computes the average of the workers' per-example gradients (Algorithm 8:6), and sends it back to all workers (Algorithm 8:7) so that they can update their local models (Algorithm 7:6-

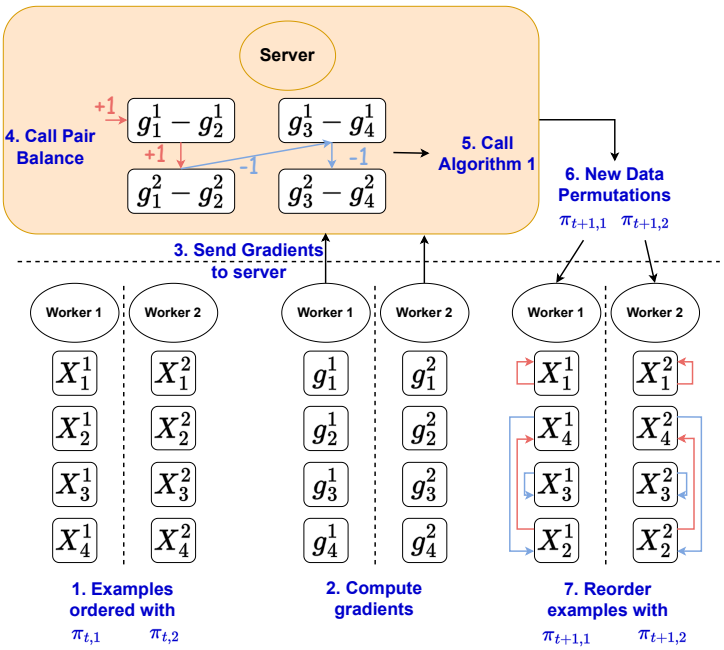


Figure 6.1: CD-GraB running on one server (top) and two workers (bottom). The workers do not share data examples. The server calls PairBalance (Algorithm 6) online.

7). Simultaneously, as the server receives gradients (Algorithm 8:5), it calls PairBalance (Algorithm 6) on adjacent vectors (4; Algorithm 8:4-13). PairBalance produces signs to supply to the reordering algorithm (Algorithm 5), which, using the current worker permutations $\pi_{t,i}$, produces the new per-worker permutations for the next epoch (5; Algorithm 8:14). In Figure 6.1, these correspond to $\pi_{t+1,1}$ and $\pi_{t+1,2}$, which the server then sends back to the respective workers (6; Algorithm 8:15). Lastly, before the start of the next epoch, the workers reorder their examples according to the new permutations (7; Algorithm 7:9).

Algorithm 7 CD-GraB Workers

require: m workers, $n \coloneqq \frac{N}{m}$ ex. per worker
input: initial $\boldsymbol{w}_1^1$, epochs T, learning rate α

receive: initial permutations $\pi_{1,i}\}_{i=1}^m$
for epoch $t \coloneqq 1 \dots T$ **do**
  ▷ Run in parallel for workers $i = 1 \dots m$
  for example $j \coloneqq 1 \dots n$ **do**
    compute: $\boldsymbol{g}_j^i \leftarrow \nabla f^i(\boldsymbol{w}_t^j, \pi_{t,i}(j))$
    send: $\boldsymbol{g}_j^i$ — j-th stochastic grad. $\boldsymbol{g}_j^i$ →
    receive: $\bar{\boldsymbol{g}}_j$ — avg. j-th stochastic grad. $\bar{\boldsymbol{g}}_j$ ←
    update: $\boldsymbol{w}_t^{j+1} \leftarrow \boldsymbol{w}_t^j - \alpha \bar{\boldsymbol{g}}_j$
  end for
  receive: next permutation $\pi_{t+1,i}$ ←
  update: $\boldsymbol{w}_{t+1}^1 \coloneqq \boldsymbol{w}_t^{n+1}$
end for
return: $\boldsymbol{w}_{T+1} \coloneqq \boldsymbol{w}_{T+1}^1$

Algorithm 8 CD-GraB PS

require: m workers, $n \coloneqq \frac{N}{m}$ ex. per worker
input: epochs T

send: initial permutations $\{\pi_{1,i}\}_{i=1}^m$
for epoch $t \coloneqq 1 \dots T$ **do**
  initialize: running sum $\boldsymbol{h} = \boldsymbol{0}$; empty list $\mathcal{S}$
  for example $j \coloneqq 1 \dots n$ **do**
    receive: $\{\boldsymbol{g}_j^i\}_{i=1}^m$ from all workers i
    compute: avg. gradient: $\bar{\boldsymbol{g}}_j \leftarrow \frac{1}{m} \sum_{i=1}^m \boldsymbol{g}_j^i$
    send: $\bar{\boldsymbol{g}}_j$ to all the workers
    for worker $i \coloneqq 1 \dots m$ **do**
      if $j \bmod 2 = 0$:
        $\boldsymbol{h}, s_{j-1}^i, s_j^i \leftarrow \mathsf{PairBalance}(\boldsymbol{h}, \boldsymbol{g}_{j-1}^i, \boldsymbol{g}_j^i)$
        $\mathcal{S}.\mathsf{append}(s_{j-1}^i)$; $\mathcal{S}.\mathsf{append}(s_j^i)$
      end for
    end for
    ▷ Call Alg. 5 for $i = 1 \dots m$ on $\pi_{t,i}$ and $\mathcal{S}$
    compute: next permutations $\{\pi_{t+1,i}\}_{i=1}^m$
    send: $\{\pi_{t+1,i}\}_{i=1}^m$ to each worker i
end for

Figure 6.2: CD-GraB worker and server (here, a parameter server [370]) algorithms.

6.4 Convergence Analysis

We next demonstrate formally that our CD-GraB algorithm (Section 6.3.3) confers the efficiency benefits of centralized GraB (Section 6.2.1) to the distributed setting. In brief, our main theoretical results show that **CD-GraB enjoys a linear speedup in convergence rate** under two sets of conditions: smoothness (Theorem 6) and the Polyak-Łojasiewicz (P.L.) condition (Theorem 7). **Both results guarantee that CD-GraB is faster than distributed random reshuffling (D-RR)**. Our proofs rely on Corollary 7 from Dwivedi and Mackey [182], which shows that, with high probability, RandomizedBalance (subroutine in Algorithm 6,

from Alweiss et al. [20]) guarantees a $\tilde{O}(1)$ bound to the signed herding objective (6.4).[8]

To begin, we restate this result to cohere with our framework, for which the vectors z_j are gradients in an optimization context:

Theorem 5 (**Corollary 7, Dwivedi and Mackey [182]**). *Consider any vectors* $\{z_j\}_{j=1}^N$ $(z_j \in \mathbb{R}^d)$ *with* $\|z_j\|_2 \leq 1$ *supplied as input to the* RandomizedBalance *subroutine in Algorithm 6. Then for any* $\delta > 0$*, with probability at least* $1 - \delta$*,* RandomizedBalance *outputs a sequence of signs* $\{s_j\}_{j=1}^N \in \{-1, 1\}$ *that satisfy* $\max_{k\in[N]} \left\|\sum_{j=1}^k s_j z_j\right\|_\infty \leq \tilde{A}$*, where* $\tilde{A} = \sqrt{2\log\left(\frac{4d}{\delta}\right)\log\left(\frac{4N}{\delta}\right)} = \tilde{O}(1)$.

To integrate this result with our parallel setting, we need some additional assumptions that are standard in the literature on distributed optimization — that the variance of the per-example gradients on each worker is uniformly bounded (Assumption 2), and that the variance between worker-specific gradients is similarly bounded (Assumption 3). More precisely, following the distributed setup in (6.7), we denote the global loss gradient to be $\nabla f(\boldsymbol{w})$, each i-th worker's local loss gradient to be $\nabla f^i(\boldsymbol{w})$ $(\forall i \in [m])$, and each i-th worker's per-example loss gradients to be $\nabla f^i(\boldsymbol{w}; j)$ $(\forall j \in [n])$. We assume:

Assumption 2 (**Bounded Grad. Var.**). *For all* $i \in [m]$ *there exists a constant* $\sigma > 0$ *such that for all* $j \in [n]$ *and for all* $\boldsymbol{w} \in \mathbb{R}^d$*, it holds that* $\left\|\nabla f^i(\boldsymbol{w}; j) - \nabla f^i(\boldsymbol{w})\right\|_2^2 \leq \sigma^2$.

Assumption 3 (**Bounded Data Heterogeneity**). *There exists a constant* $\varsigma > 0$ *such that* $\forall i \in [m]$*,* $\left\|\nabla f^i(\boldsymbol{w}) - \nabla f(\boldsymbol{w})\right\|_2^2 \leq \varsigma^2$.

Lastly, we include one additional assumption from the original GraB pa-

[8] Corollary 7 from Dwivedi and Mackey [182] improves the result of Theorem 1.1 from Alweiss et al. [20].

per [384]: we assume a cross norm $L_{2,\infty}$ (which can be easily adapted to L_2-smoothness by setting $L_{2,\infty}$ to be $\sqrt{d}L_2$).

Assumption 4 (Smoothness). *There exists constant $L_{2,\infty} > 0$ such that for any $\boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^d$, any $i \in [m]$, and any $j \in [n]$, it holds that $\left\|\nabla f^i(\boldsymbol{w}; j) - \nabla f^i(\boldsymbol{v}; j)\right\|_2 \leq L_{2,\infty}\|\boldsymbol{w} - \boldsymbol{v}\|_\infty$.*

Given these assumptions, we can prove a convergence guarantee for CD-GraB:

Theorem 6. *Suppose that Assumptions 2,3 and 4 hold. For any $\delta > 0$, if we set learning rate α to be*

$$\alpha = \min\left\{\frac{1}{16L_{2,\infty}(2n + \tilde{A}/m)}, \left(\frac{4F_1m^2}{42L_{2,\infty}^2(\varsigma + \sigma)^2\tilde{A}^2nT + 18L_{2,\infty}^2m^2n^3\sigma^2}\right)^{1/3}\right\},$$

where $F_1 = f(\boldsymbol{w}_1) - \inf_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w})$ and $\tilde{A}$ comes from Theorem 5. Then, with probability at least $1 - T\delta$,

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\boldsymbol{w}_t)\|_2^2 &\leq \frac{9(F_1L_{2,\infty}(\varsigma + \sigma)\tilde{A})^{2/3}}{(mnT)^{2/3}} + \frac{(72F_1L_{2,\infty}\sigma)^{2/3} + 64F_1L_{2,\infty}(2 + \tilde{A}/(mn))}{T} \\ &= \tilde{O}\left(\frac{1}{(mnT)^{2/3}} + \frac{1}{T}\right).\end{aligned}$$

We can also prove an accelerated rate for CD-GraB if we additionally assume the P.L. condition:

Assumption 5 (P.L. Condition). *We say the loss function f fulfills the P.L. condition if there exists $\mu > 0$ such that for any $\boldsymbol{w} \in \mathbb{R}^d$, $\frac{1}{2}\|\nabla f(\boldsymbol{w})\|_2^2 \geq \mu(f(\boldsymbol{w}) - \inf_{\boldsymbol{v}\in\mathbb{R}^d} f(\boldsymbol{v}))$.*

Theorem 7. *Suppose that Assumptions 2, 3, 4, and 5 hold. For any $\delta > 0$, we set constants $\tilde{W}$ and C_3 to be*

$$C_3 = \frac{(F_1 + \sigma^2/L_{2,\infty})\mu^2}{224L_{2,\infty}^2(\varsigma + \sigma)^2\tilde{A}^2} \quad \text{and} \quad \tilde{W} = W_0(T^2m^2n^2C_3),$$

where $\tilde{A}$ comes from Theorem 5, F_1 is from Theorem 6, and W_0 is the Lambert-W function. If we set learning rate $\alpha = \frac{2\tilde{W}}{Tn\mu}$ and if the number of epochs T satisfies

$$T \geq 10 + \frac{1}{\mu} 32 L_{2,\infty}(2 + \tilde{A}/(mn)) W_0((mnT)^2 C_3) = \tilde{O}(1),$$

then, with probability at least $1 - T\delta$, it holds that

$$F_{T+1} \leq \frac{1}{(mnT)^2} \left(\frac{(F_1 + L_{2,\infty}^2 \sigma^2)\tilde{W}}{C_3} + \frac{112 L_{2,\infty}^2 (\varsigma + \sigma)^2 \tilde{A}^2 \tilde{W}^2}{\mu^3} \right) = \tilde{O}\left(\frac{1}{(mnT)^2} \right),$$

where $F_{T+1} = f(\boldsymbol{w}_{T+1}) - \inf_{\boldsymbol{w} \in \mathbb{R}^d} f(\boldsymbol{w})$.

We prove Theorems 6 and 7 in the Appendix (Chapter E). Together, they show that CD-GraB exhibits a linear speedup in the number of workers m over GraB [384]'s convergence rates ($\tilde{O}((nT)^{-2/3})$ and $\tilde{O}((nT)^{-2})$, respectively).[9] under both smoothness and the P.L. condition. Further, CD-GraB's convergence rate of $\tilde{O}((mnT)^{-2})$ is faster than many previous rates,[10] such as the high probability bound of $\tilde{O}((mn)^{-1}T^{-2})$ for D-RR in Yun et al. [638].

6.5 CD-GraB in Practice: Distributed and Simulation Experiments

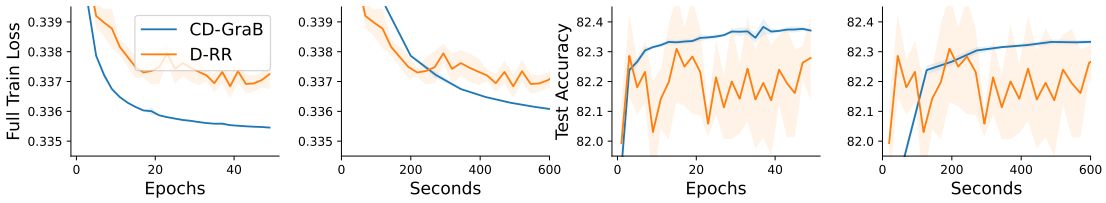
We next verify CD-GraB's accelerated convergence on a variety of empirical tasks.[11] For ease of comparison, we follow the experimental plan from the original GraB paper,[12] and add some additional large-scale logistic regression

[9] For centralized GraB, the total number of examples $N = n$ and $m = 1$.

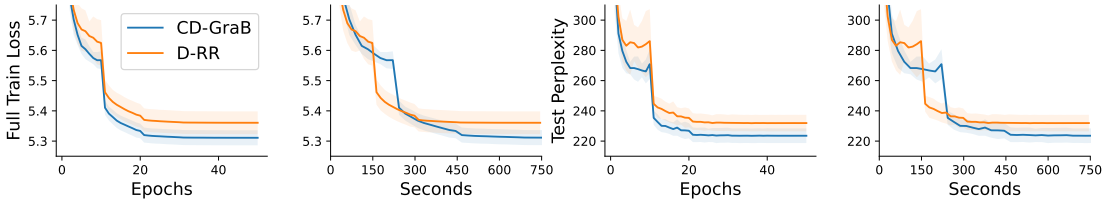
[10] These exclusively focus on the P.L. case, so we compare CD-GraB to them under the same condition.

[11] Our GitHub repository is https://github.com/GarlGuo/CD-GraB.

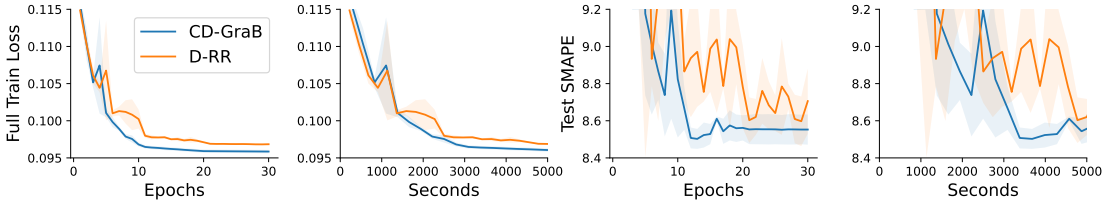
[12] Following Lu et al.[384], for our LSTM experiment on WikiText-2, we set the embedding dimension to 32. We note that we can improve perplexity if we set the dimension higher.



(a) Logistic regression on mortgage application (NY 2017 subset) [141]



(b) LSTM [280] on WikiText-2 [407]



(c) Autoregressive MLP on M4 Weekly [391]

Figure 6.3: Convergence of CD-GraB in comparison to D-RR. For each experiment, we show train loss over epochs and time (**left** of each subfigure) and test performance over epochs and time (**right** of each subfigure). We run at least 3 random seeds, and plot the mean ± STD.

experiments. We also run an ablation study to isolate the effects of different improvements in CD-GraB. We do this because online PairBalance exhibits performance benefits that are separate from parallelism — namely, removing the need for gradient centering with a stale mean and allowing for higher learning rates (Section 6.3.2).[13]

Evaluating CD-GraB's convergence speedup. We use the following three tasks for evaluating distributed training efficiency: logistic regression on a large-scale mortgage application (New York 2017 subset, 244,107 examples with 18 features) [141] (Figure 6.3a), Long Short-Term Memory (LSTM) [280] on the WikiText-2 dataset [407] (Figure 6.3b), and autoregressive Multi-Layer Percep-

[13]GraB can also implement online PairBalance, in place of Balance [385] (Appendix).

tron (MLP) on the M4 Weekly dataset [391] (Figure 6.3c). We measure the loss incurred on the entire training set (Full Train Loss) and task-appropriate test metrics during evaluation, with respect to both the number of epochs and wall-clock time. Regarding test metrics, we measure test accuracy for the mortgage application, perplexity for WikiText-2, and SMAPE for M4. Additional details regarding the datasets, models, and test metrics can be found in the Appendix (Chapter E).

For all three tasks, we use a single 128 GiB memory machine with 4 NVIDIA GeForce RTX 2080 Ti GPUs. For the mortgage application and WikiText-2 (Figures 6.3a and 6.3b), we launch $m = 4$ workers (processes), where each worker runs on one GPU. For the M4 task, we launch $m = 32$ workers, where each of the 4 GPUs hosts 8 process workers. We use NCCL as the distributed communication backend [447] for the mortgage application and WikiText-2 tasks, and GLOO [292] as the distributed communication backend for the M4 task.

As shown in Figure 6.3, we compare CD-GraB's convergence to the standard distributed-training example-ordering method: random reshuffling (D-RR). From all subfigures in Figure 6.3, we observe that CD-GraB outperforms the D-RR baseline significantly and consistently: CD-GraB exhibits better training loss and test metrics, measured against both the number of epochs and wall-clock time. We also note that the results for CD-GraB are much smoother than for D-RR. This is likely due to the variance of stochastic gradients during training, which CD-GraB reduces as a side-effect (so, too, does GraB, in comparison to RR). For smoother D-RR results, we can reduce the learning rate (Appendix, Chapter E). CD-GraB allows for the use of a larger learning rate, which accelerates training while preserving the final model's performance.

Ablation simulation study: the importance of coordination at large scale. CD-GraB has several design benefits over the original centralized GraB algorithm [384]: coordinating parallel workers' specific permutations using PairBalance on the server (Algorithm 6.2) and removing the dependency on a stale mean (Section 6.2.1), which enables the ability to using larger learning rates reliably (Section 6.3.2). Clearly, not all of these benefits come directly from distributing training. For example, being able to use larger learning rates, is a side effect of our solution to develop CD-GraB, not our main contribution. Therefore, we run a simulation ablation study to disentangle the relative importance of each of CD-GraB's efficiency benefits over GraB. To do so, we compare the convergence of CD-GraB to two additional baselines in the distributed setting, beyond D-RR: (1) **ID-GraB (Bal)**, where each independent worker runs GraB locally using RandomizedBalance (subroutine in Algorithm 6) to perform gradient vector balancing; (2) **ID-GraB (PairBal)**, where each independent worker runs GraB locally using PairBalance.

Figure 6.4 summarizes the results, with convergence curves for $m \in \{4, 8, 16, 32, 64\}$ workers training LeNet on CIFAR-10. We choose this task and architecture to cohere with the experiments done in the original GraB paper. For these experiments, we denote B to be the *aggregated* minibatch across all the workers, which refers to the number of stochastic examples used for an overall optimization step; each worker thus has a subset of this minibatch — an equivalently-sized subset of B examples.[14] We make two main observations. First, when scaling up training with more workers, CD-GraB converges increasingly faster than the no-coordination-ordering methods **ID-GraB (Bal)** and **ID-**

[14]For example, if we have 4 workers with an aggregated minibatch size of 32, each worker would compute their respective local gradients with 8 examples, and then all-reduce these gradients to obtain the aggregated minibatch gradient for all 32 examples for the optimization step. We discard N mod B examples at random to ensure n examples per worker.

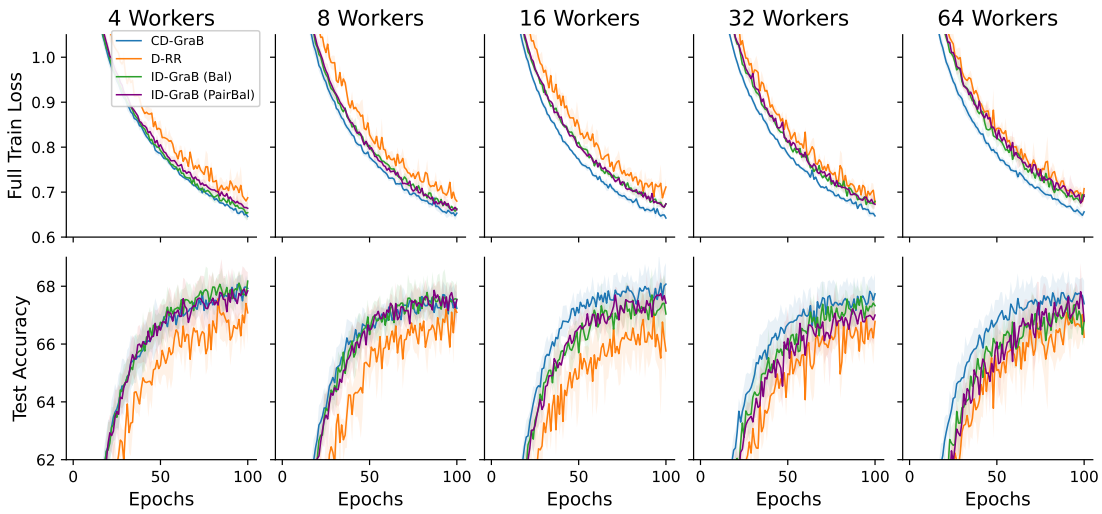


Figure 6.4: Convergence for CD-GraB, D-RR, ID-GraB (Bal), and ID-GraB (PairBal) training LeNet on CIFAR-10, with $m \in \{4, 8, 16, 32, 64\}$ workers. For each experiment, the aggregated minibatch size per update is 64.

GraB (PairBal). This result aligns with our theory and intuition that, when the number of workers m increases, the parallel herding bound (6.8) will increase linearly if there is no coordination. Second, as we scale up to larger m, the convergence curves of **ID-GraB (Bal)** and **ID-GraB (PairBal)** gradually approach the curve for D-RR: at larger scales, herding-based example ordering will be no better than randomly permuting the dataset. Both observations give strong evidence that coordination (i.e., running online PairBalance on the server to coordinate per-worker permutations) is critical for accelerating training.

We note that all of these experiments use SGD, since both the theoretical results of the original GraB paper and our results for CD-GraB here are for SGD. In the Appendix (Chapter E), we additionally include results for training GPT-2 on WikiText-103, for which we use AdamW as the optimizer. We find that GraB with AdamW works in practice; however, our theory results do not directly apply to these experiments. We additionally include results on memory usage in the Appendix, which show that CD-GraB results in negligible overhead in

practice.

6.6 Conclusion and Future Work: Toward an Order Server Architecture

We elevate the benefits of provably faster, permutation-based example ordering to the contemporary ML distributed-training setting. We focus on reformulating the online **Gra**dient **B**alancing algorithm (GraB) [384] because, even though it is the provably optimal permutation-based example-ordering method [115], it is limited by design to *centralized* settings (Section 6.3.1). To overcome these limitations, we redesign GraB's herding and balancing framework to account for parallel workers: A *parallel herding* objective, which we solve with an online PairBalance subroutine, based on key insights from kernel thinning [47, 182, 183]. PairBalance operates on ordered *pairs* of vectors to do *balancing*, which enables our full-stack, low-overhead, ***C**oordinated* and ***D**istributed* online CD-GraB algorithm. We give a full specification of our online CD-GraB algorithm (Section 6.3.3), provide convergence rate guarantees regarding its speedups on both 1) smooth non-convex and 2) P.L. objectives (Section 6.4), and verify these speedups in practice on single-node distributed tasks and a simulated ablation study (Section 6.5).

Both our theory and experiments demonstrate that CD-GraB really shines when there are multiple training epochs (Appendix). This is another reason that we do not emphasize experiments involving fine-tuning pre-trained models like GPT-2, as fine-tuning can be achieved in just a couple of epochs. As noted above, it is also more common to train such models using optimizers from the Adam

family. In future work, we intend to extend the theory on GraB and CD-GraB to such optimizers, which would make the results on optimal, permutation-based example ordering more useful for base-model pre-training.

Pre-training from scratch would demonstrate the tremendous power of CD-GraB to scale to very large models; however, we did not have the training budget to perform such experiments for the present work. Further, to truly exercise the benefits of CD-GraB in such large-scale settings, future work should investigate moving beyond the single-node setup that we present. Notably, to train larger models, our results suggest a novel distributed training architecture. The ordering operation performed by the server (Algorithm 8) is *not* very latency sensitive; the server has the duration of the entire epoch t to compute the new permutations for the next, $t + 1$ epoch. Given this relaxed latency requirement, and the success of our algorithmic results, it would be an exciting direction for future ML-systems research to invest in building an *Order Server* architecture. Such an architecture, which could be composed with traditional parameter servers, would afford the scalability benefits of CD-GraB to a host of massive-scale ML applications.

CHAPTER 7

ACCURACY-EFFICIENCY TRADE-OFFS AND ACCOUNTABILITY

Machine-learning algorithms that attempt to afford greater scalability and efficiency tend to trade-off these gains for a reduction in (overall or per-iteration) accuracy. Indeed, such accuracy-efficiency trade-offs (and how to wrangle them) lay at the heart of scalable machine learning. In this chapter, we connect our algorithmic work from scalable machine learning, presented in the two prior chapters, to insights in law and policy. We explore how the accuracy-efficiency trade-offs inherent to scalable machine learning also provide a natural level of abstraction for reasoning about law and policy implications.

This work, initially published before the advent of "Generative AI," has aged surprisingly well. The motivating example of a distributed ML system in this text is an autonomous vehicle (AV) — IoT and AVs were (at the time of writing) the expected application space where our observations would be relevant. While this has not come to pass (there is not yet widespread deployment of AV systems), we have seen other large-scale, distributed ML systems become household topics, e.g., ChatGPT. The observations and arguments in this paper are largely applicable to such systems. We deffer this unification to future work.

Chapter summary: Trade-offs between accuracy and efficiency pervade law, public health, and other non-computing domains, which have developed policies to guide how to balance the two in conditions of uncertainty. While computer science also commonly studies accuracy-efficiency trade-offs, their policy implications remain poorly examined. Drawing on risk assessment practices in the US, we argue that, since examining these trade-offs has been useful for guiding governance in other domains, we need to similarly reckon with these

trade-offs in governing computer systems. We focus our analysis on distributed machine learning systems. Understanding the policy implications in this area is particularly urgent because such systems, which include autonomous vehicles, tend to be high-stakes and safety-critical. We 1) describe how the trade-off takes shape for these systems, 2) highlight gaps between existing US risk assessment standards and what these systems require to be properly assessed, and 3) make specific calls to action to facilitate accountability when hypothetical risks concerning the accuracy-efficiency trade-off become realized as accidents in the real world. We close by discussing how such accountability mechanisms encourage more just, transparent governance aligned with public values.

This chapter is a licensed derivative copy of work published and awarded an oral presentation slot at *ACM EAAMO 2021* [144]. Another version of this work, targeted at a legal audience, was published in the *Colorado Technology Law Journal* in 2022 [143].

7.1 Introduction

Engineering is defined by trade-offs — by competing goals that need to be negotiated in order to meet system design requirements. One of the central trade-offs, particularly in computer science, is between *accuracy* and *efficiency*. There is an inherent tension between *how correct* computations are and *how long* it takes to compute them. While this trade-off is of general relevance, it plays out in various ways across computing: in computer hardware, circuits can use approximation techniques to relax constraints on accuracy — on how they perform bitwise computations — to speed up performance; in image processing, compressing pixels causes a loss in accuracy of the image being represented,

but also furthers space-efficiency by requiring less memory for storage. In fact, such trade-offs are so abundant in computing that they have even given rise to its own subfield, *approximate computing* [418, 421], which studies how different domains resolve the question of how much inaccuracy can safely be permitted for the sake of increased efficiency [511].

While the trade-off is commonly acknowledged in computer science, its policy implications remain poorly examined. We provide a starting point, in which we focus our analysis on *distributed ML systems* using the running example of autonomous vehicles (AVs). We make this choice for two reasons. The first is urgency: AV development has made such significant strides that by 2040 at least 75% of cars will have some level of autonomy [439]. Second, while AVs promise to improve overall driving safety,[1] they will also create new risks [78, 458]. As we show, some of these risks directly result from the accuracy-efficiency trade-off and the choices made to implement it [437]. In particular, the trade-off is tunable and context-dependent: it is not an all-or-nothing choice, and appropriate tuning depends on both a system's goals and deployment environment. Choices in different contexts will entail different emergent behaviors in technical systems — behaviors that are potentially high-stakes if, for example, they affect overall system safety.

We argue that the accuracy-efficiency trade-off exposes a high-level abstraction that policymakers should use to help hold such systems accountable.[2]

[1]The international effort to deploy AVs is motivated in large part due to AV technology's promise to increase automotive safety — that replacing human drivers with automated ones will protect millions of lives. Conservative estimates indicate that in 2035-2045, the decade in which AVs are targeted to reach widespread deployment, 585,000 lives will be saved worldwide [498].

[2]We emphasize that this is *not the only* such tool policymakers should have for holding these systems accountable. Other accountability mechanisms are also necessary, such as those that can assess hardware failures [6, 22, 566], the explainability of ML models [334], and the impact of variance in automated decision-making [210].

Rather than operating at one of two extremes — solely having policymakers rely on technical experts to make high-stakes decisions or inundating policymakers with underlying low-level technical details — we advocate for something in between: researchers should focus on providing correctness and performance guarantees, and should build tools to help policymakers reason about these guarantees. These tools should help expose the uncertainty in distributed ML systems. This would facilitate lawmakers' ability to assess whether tradeoff implementations are aligned with safety goals, and to regulate the risk of deploying high-stakes systems like AVs.

We emphasize *distributed systems* because much of the sociotechnical conversation in ML has focused on *algorithmic* fairness. This has left the systems components — notably, scalability, speed and their impact on correctness — under-explored in terms of their policy implications. As a result, ML *systems* present under-examined challenges for technological accountability. We take the initial steps to bring some of these challenges to light, and suggest a novel framing for how to hold such systems accountable.

This contribution demonstrates the need for mandatory risk assessment tools for distributed ML systems. We contend that, without such tools, effective public oversight of these systems will not be possible. Instead, we run the risk of manufacturers ignoring accountability mechanisms when constructing ML systems — or worse, deliberately making these systems difficult to assess in order to obscure responsibility when accidents occur. In both of these scenarios, the burden would fall on individual victims to prove manufacturer responsibility. This dynamic would make accountability quite difficult to achieve; the power and resource imbalances between individual victims and large ML-

system manufacturers would make tort or other civil litigation infeasible [6].

Our analysis focuses on the US, but elicits principles that apply more broadly. We have chosen AVs as our central example because navigating the trade-off appropriately has already proven an urgent concern, notably in assessing Uber's 2018 AV crash [437]. To make our case, we survey relevant concepts and examples from law and computer science, and then synthesize this discussion to advocate for a concrete policy contribution, which we direct toward the National Highway Transportation Safety Authority (NHTSA).[3] We first discuss how the trade-off functions in relation to decision-making in disciplines other than computing, most notably in US risk assessment policy (Section 7.2). Then, we provide an analogous discussion for ML algorithms and distributed ML systems (Section 7.3). We argue that reasoning about accuracy-efficiency trade-offs and accountability in highly technical domains is not a new problem. This suggests that, with the right technical tools, we can similarly hold high-stakes, distributed ML systems like AVs accountable (Section 7.4) with respect to how they implement analogous trade-offs. We close this chapter by discussing how such tools for increased accountability encourage more just, transparent governance aligned with public values (Section 7.5).

7.2 The Ubiquity of Accuracy-Efficiency Trade-Offs

The trade-off at the heart of this paper is not unique to computing. It can be observed in a range of domains, many of which are regulated in the US, including

[3] Approaching our topic in this interdisciplinary manner leads us to follow a nontraditional format. We need to justify our conceptual contribution in two directions, and thus provide a significant amount of relevant background information concerning how the accuracy-efficiency trade-off translates to both law and computer science.

law, the economy, and public health.[4] In these disciplines, efficiency often can be thought of interchangeably with speed. For example, in decision theory, the time-value of information is an important concept for making choices. There is a cost to gathering increasingly accurate information: waiting to act is itself an action — one that can have more negative consequences than acting earlier on imperfect information[5]

Sunstein [564] connects this idea to the potential hazards of using heuristics in legal decision-making. Nevertheless, he observes that heuristics are common (and necessary) to obtain a suitable balance between efficient resolution and the "best" (i.e., most accurate) adjudicative outcomes.[6] For example, a number of rules in US civil and criminal procedure — speedy trial requirements, local filing deadlines, statutes of limitations — impose time constraints for the sake of efficient case resolution; these values must be balanced against needs for thorough fact-finding and argumentation. The standard for preliminary injunctive relief in the US requires courts to predict whether irreparable injury will occur because of the passage of time, if relief is not granted before the (often lengthy) full resolution of a case [373]. Federal Rule of Evidence 403 allows for the exclusion of relevant evidence from a court proceeding if the probative value of that evidence is substantially outweighed by a danger of undue delay. These and other rules promoting judicial efficiency are, in the words of Justice Oliver

[4]The accuracy-efficiency trade-off is also salient in other aspects of governance, including wartime intelligence gathering. The "fog of war" concerns the inherent tension between gathering more accurate intelligence about an opponent or enemy and acting on that intelligence before it becomes stale and loses its usefulness [604].

[5]Kahneman et al. elaborates on this idea in well-known cognitive psychology research concerning reasoning about uncertainty [306] The authors argue that humans use various heuristics to make decisions more efficiently, often acting on biases they have due to incomplete information. There is a tension between taking the time to gather more information and making a more informed decision — between the speed of making a decision and the quality of information used to make it.

[6]Due process is perhaps the most notable, encompassing example of balancing both values in US law.

Wendell Holmes, "a concession to the shortness of life" [589] — they attempt to balance between the twin goals of getting matters right and getting them done, with recognition that there is real social value to each.

Debates about the merits of the "precautionary principle" in policymaking also reflect the trade-off. The precautionary principle advises extreme caution around new innovations when there is substantial unknown risk; it places the burden of proof on risk-creating actors (like chemical plants) to provide sufficient evidence that they are *not* producing significant risk of harm. As with speedy trials, there is a trade-off between the time it takes to gather evidence — to understand the risk landscape — and making informed decisions based on this landscape.[7]

A notable example of the precautionary principle demonstrating the trade-off in action concerns public health management of the SARS outbreak in the early 2000s. During the early outbreak of the disease, there was significant uncertainty around the risk of it spreading and how lethal it could be. The principle was adopted as a public health value at all of the disease epicenters: individuals who were even remotely suspected of having come into contact with SARS were placed under strict quarantine. Years later, (pre-COVID-19) critics argued that mass quarantining led to a tremendous and unnecessary loss of liberty. They made this case based on analysis that indicated 66% fewer individuals could have been quarantined with the same public health outcome (i.e.,

[7]There are legal rationales on both sides of the spectrum with regard to how this trade-off should be implemented. For example, critics of the precautionary principle could be said to favor efficiency. They find the principle to be too stringent with regard to the burden it places on accuracy; it is "literally paralyzing" in its attempts to regulate risk [565]. On the other side, others argue that the precautionary principle provides a valuable way to reason about preventing harm by shifting the burden of proof of safety to potential risk creators. They are supportive of the fact that the principle requires actors to justify the risks they create: it is worth the time cost to gather information, such that it is possible to better manage risk in the context of scientific uncertainty [506].

it would have still been possible to prevent a SARS pandemic) [124].[8]

7.2.1 US federal risk assessment policy

The examples above provide an intuition for how pervasive the accuracy-efficiency trade-off is in different domains, and how it is reasoned about to guide decision-making. Beyond this intuition, the trade-off is implicated more formally in US federal risk assessment standards and regulatory rule-making. Risk assessment policy acknowledges that, no matter how much time and resources one spends gathering scientific knowledge to assess risks, it will ultimately always be necessary to make decisions with uncertainty — to pass judgments in the face of incomplete information [157, 158].[9] There is always a degree of imprecision in scientific knowledge's ability to capture what is true, and that knowledge is constantly subject to revision in light of newly collected information. That is, taking more time to gather information can increase accuracy, but is directly at odds with efficiency in decision-making.

[8] We are not yet at a time in which such retrospective analysis regarding the precautionary principle can be conducted for the ongoing COVID-19 pandemic. Nevertheless, the trade-off has still played a role in an additional public health context: antibody tests. The World Health Organization (WHO) has recently argued that, prior to certifying COVID-19 antibodies for treatment, it is necessary to *guarantee* that such antibodies confer immunity to the virus. Several medical professionals have challenged this mandate from WHO, highlighting the time-sensitive nature of taking action in the pandemic: "Demanding incontrovertible evidence may be appropriate in the rarefied world of scholarly scientific inquiry. But in the context of a raging pandemic, we simply do not have the luxury of holding decisions in abeyance until all the relevant evidence can be assembled. Failing to take action is itself an action that carries profound costs and health consequences." More generally, it is the norm for healthcare practitioners to act on incomplete information — to balance potential inaccuracies in available data with the urgency to treat serious conditions [619].

[9] As Levy and Johns notes, it is the epistemological nature of science itself that makes uncertainty inevitable in science-based policymaking: "Agencies charged with protecting public health and the environment must make decisions in the face of scientific uncertainty, because science by its nature is incomplete and only rarely provides precise answers to the complex questions policymakers pose" [367].

In risk assessment, this trade-off is framed in terms of *ex ante* (before-the-fact) and *ex post* (after-the-fact) risk-mitigating interventions. The AI safety and fairness communities sometimes use the terms *assessment* and *audit*, respectively for *ex ante* and *ex post* [196]. *Ex ante* mechanisms embody the precautionary approach: they emphasize collecting evidence about potential risks before approving a new substance or technology. For example, the FDA[10] typically requires multiple phases of clinical trials before a new drug is approved for use (i.e., "premarketing approval" [10, 158]). This *ex ante* regulatory authority is deliberately slow for the sake of increased safety.[11]

In contrast, for efficiency, other agencies concentrate their authority in *ex post* "post hoc mechanisms" [158].[12] NHTSA has relatively weak *ex ante* authority for determining what types of vehicles are safe to drive; its strongest authority is the ability to recall faulty cars *ex post* [10, 601].[13] NHTSA favors lack of *ex ante* regulation as a way to ensure speedy development and deployment of new car technology, even if such lack of regulation comes with a cost in correctness in that technology. These are just two examples illustrating opposite choices concerning how accuracy and efficiency relate to *ex ante* and *ex post* enforcement. This trade-off spectrum applies to the risk assessment and rule-making practices

[10]US Food and Drug Administration (FDA).

[11]The FDA is empowered to require drug companies to submit sufficient data, such that a detailed risk assessment can be conducted before the drug goes on the market. This process can take a lot of time, and is not always conducted without criticism concerning choosing "safety" over "efficiency". For example, such critiques are common when swift approval has known safety benefits, but is delayed in favor of evaluating the presence of unknown (potentially non-existent) health risks. Debates concerning the FDA and this accuracy-efficiency trade-off have been particularly relevant recently concerning approving COVID vaccines for children [461].

[12]These mechanisms tend to require that agencies, rather than companies, acquire the data necessary to determine responsibility after an undesirable outcome occurs.

[13]NHTSA has the ability to set safety standards, and then verifies that manufacturers have met them through a self-certification process. In other words, manufacturers certify themselves as "safe," rather than NHTSA soliciting data from manufacturers and performing the certification themselves [10, 601].

of numerous other US agencies, including the EPA,[14] OSHA,[15] and the CPSC,[16] which each have different, domain-specific *ex ante* and *ex post* biases.

Despite these differences, reports from the NRC[17] recognize that there are cross-cutting elements of risk assessment [157, 158]. The reports provide general recommendations for improving standards for accounting for uncertainty and its relationship to risk, such as clarifying the assumptions that inform model construction to elucidate model uncertainty. The NRC advocates for the importance of teasing out these low-level details, and communicating them to both decision-makers and the public, in order to ensure that policy goals reflect the known risk landscape.

This discussion shows that accuracy-efficiency trade-offs are a useful and natural way for policymakers to regulate varied, complex technical domains. We therefore ask: why not use this framework for making policy concerning distributed ML systems? The specifics of the domain may vary — notably, real-time systems involve high speeds not present in, for example, evaluating the safety of new chemicals. Nevertheless, US risk assessment policy indicates that reasoning about accuracy-efficiency trade-offs, and their relationship to risk, is not a new problem. We therefore contend that reasoning about underlying accuracy-efficiency trade-offs can enable risk assessment and management for these emerging technologies. However, translating the above regulatory framing to this domain presents novel challenges. We will require new tools, which we clarify in Sections 7.3 and 7.4, to reason effectively about similar trade-offs in distributed ML systems — tools that expose the particular type of uncertainty

[14]Environmental Protection Agency (EPA).
[15]Occupational Safety and Health Administration (OSHA).
[16]Consumer Product Safety Commission (CPSC).
[17]National Research Council (NRC).

in real-time, distributed, automated decision-making. These tools will help us gather the data necessary for appropriate risk assessment and policymaking.

Before we can describe these tools, we clarify that accuracy-efficiency trade-offs are an appropriate abstraction for accounting for the behavior of distributed ML systems. Having explained how reasoning about such trade-offs is useful for policymaking, we next make our case from a technical perspective.

7.3 Trading off Accuracy and Efficiency in Computing

Accuracy-efficiency trade-offs are particularly relevant across computing.[18] To understand this, consider a familiar example — JPEG compression. Raw images tend to be very high resolution: they contain many, varied pixels per inch, and therefore require a lot of storage space. However, a compressed, JPEG version often suffices for high quality; combining neighboring pixels often is not detectable to the human eye. A JPEG also takes up less storage space and can lead to faster processing when doing photo editing since there are fewer pixels to consider; it is more space- and time-efficient. Reducing the accuracy of the image can lead to greater computational efficiencies. This type of trade-off spectrum forms the basis of *approximate computing* (Figure 7.1), which studies how a computer system can achieve certain performance benefits if it exerts less computational effort to compute perfectly accurate answers. In

[18]The accuracy-efficiency trade-off is arguably a central concern for the entire field of computing. Ohm and Frankle call efficiency the "cardinal virtue" of computing in order to discuss what they view as exceptional cases of inserting inefficiency into computer systems — what they term "desirable inefficiency" [448]. Instead, viewing the accuracy-efficiency *trade-off* as central enables us to not refer to "inefficient" computing models (e.g. cryptography) as exceptional. We conceive of them as implementing the trade-off at one end of the accuracy-efficiency spectrum (with cryptography privileging accuracy), which strikes us as a more precise and generalizable statement.

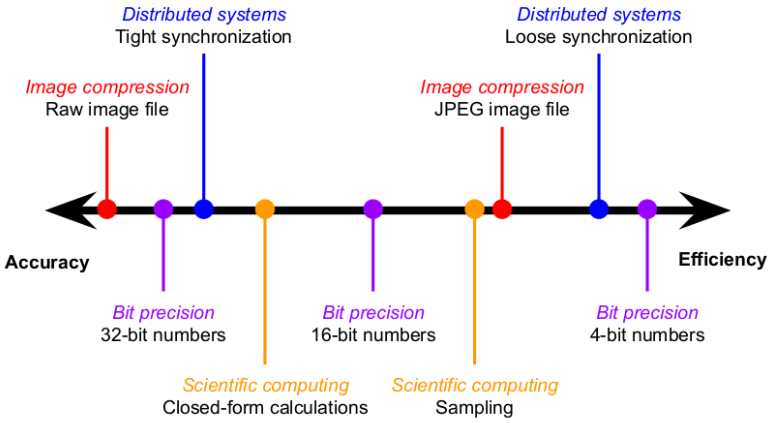


Figure 7.1: Computing examples of the accuracy-efficiency trade-off spectrum: Image compression (raw images are higher accuracy; JPEGs are more efficient), bit precision (32-bit numbers are higher accuracy; 8-bit numbers are more efficient; 16-bit numbers reflect an in-between), distributed systems (tight synchronization is higher accuracy; loose synchronization is more efficient), and scientific computing (closed-form solutions are higher accuracy; sampling is more efficient). There are diminishing returns toward either end of the spectrum.

other words, it is possible to *relax* accuracy in order to yield efficiency improvements [418, 421, 511].[19]

As with JPEGs, relaxing accuracy does not necessarily have negative consequences; rather, it is possible that decreased accuracy has no observable impact for a particular application. That is, some applications are tolerant of inaccuracy; they are error resilient. Similar to non-computing domains, tools for reasoning about the trade-off inform decisions about how to implement it. Computer scientists create theoretical tools to characterize the trade-off, which they leverage to determine the right implementation in different applications. Formal reasoning about the trade-off can yield application-specific quality metrics,

[19]We do not include the pathological case in which *all* accuracy is sacrificed in order to do something really fast but completely wrong. Nevertheless, there are cases where an implementation could, for example, be wrong 40% of the time (for increased speed) and still achieve certain application-specific quality goals.

where quality can be thought of as whether a program produces "good enough" results. Often, "good enough" cannot be guaranteed with complete certainty, but can be verified with high probability. Leaving room for uncertainty allows for edge case behaviors that fall below the specified quality threshold. Quality metrics therefore capture how much an approximation is allowed to deviate from the precise version's results. Computer scientists can then design software that requires a certain degree of program quality with a certain (high) probability [511].[20]

7.3.1 Accuracy-efficiency trade-offs in ML

Such trade-offs are a salient concern across ML. Notably, in deep learning, there is an ongoing, increasing emphasis on training larger models to yield more accurate results. This comes with host of efficiency challenges, including significantly increased training time, model storage requirements, and energy usage [310].[21] Moreover, ML models perform inference that is not always correct; to be robust, models need to tolerate a certain degree of inaccuracy. This notion of error resilience (or inaccuracy tolerance) varies for different ML algorithms. Regardless of particular differences, there is a general tension between *correctness* and *performance*.[22] In fact, relaxing accuracy to increase efficiency is a re-

[20] A practical example of this comes from Amazon's cloud computing services (AWS). Their cloud storage service provides "11 9's" of reliability with regard to storing data objects, meaning that 99.999999999% of the time saving such objects to the cloud occurs without error [21].

[21] The trade-off notably did not first become relevant with (though is arguably increasingly urgent due to) the advent of modern statistical ML. Several influential papers on artificial intelligence (AI) from the 1980s and 1990s also demonstrate the potentially high impact of appropriately dealing with accuracy-efficiency trade-offs [72, 281].

[22] For example, the correctness of a training algorithm can be understood as whether or not the algorithm converged to the distribution we set out to learn, i.e., *Did we learn the right model?* Its performance indicates whether convergence to the distribution — whether correct or incorrect — happened in a timely manner, i.e., *How fast did we learn the model?*

quirement in many learning domains. Otherwise, computations can be so slow to perform that they become intractable.

One relaxation strategy[23] is *subsampling* during training, which involves using a subset of the dataset in place of the entire dataset to compute model updates faster.[24] Even though each iteration is less accurate (but more efficient), some algorithms can still guarantee overall high-quality (i.e., statistically correct) results. A very common approach for improving efficiency is to use a subsample or *minibatch* of the dataset, rather than the whole dataset, when performing calculations. In the case of computing gradients, instead of using a *full batch* (i.e., the whole dataset) we use a randomly sampled subset of the data points, which involves spending less time on the computation of a particular iteration.[25]

A second strategy is *asynchrony*, which enables different computer processes

[23]We give four general strategies in this paper, which are far from exhaustive. Notable examples of subfields with specific trade-offs include reinforcement learning (RL) and Markov chain Monte Carlo (MCMC). In RL, there is the well-known exploration-exploitation trade-off (more exploration increases accuracy and more exploitation increases efficiency) [296, 300]. In MCMC, algorithms exhibit scalability-reliability trade-offs (scalability corresponds to efficiency, reliability to accuracy) [641].

[24]Performance directly relates to the size of the task on which we conduct learning. Intuitively, if a learning algorithm is slow on tasks with small datasets, then that algorithm will be slow, if not computationally intractable, on much larger ones. This relationship between runtime and task size often exists due to coupling between the computation done by the learning procedure's optimization algorithm and the task's dataset size. For example, when computing the gradient needed to determine which direction the learning algorithm should step for its next iteration, it is often necessary to sum over every data point in the dataset.

[25]Stochastic Gradient Descent (SGD) is an example of an algorithm that takes this approach, in which using a minibatch can have minimal impact on the overall accuracy of the learned model. A particular iteration of the algorithm will have less accuracy when computing the gradient; but, when run for lots of iterations, the final result is usually still statistically correct. In expectation, we can learn the same distribution as if we had been using the whole dataset in each iteration; we can often theoretically guarantee robustness [76]. Moreover, the decision to subsample is not all-or-nothing; it is a spectrum. It is possible to vary the minibatch size the algorithm uses. Larger minibatches — especially those that approach the size of the full dataset — require more time but are also more accurate per iteration. Conversely, smaller batch sizes make each iteration faster and more scalable to larger datasets, but in doing so sacrifice accuracy per iteration. Determining the right sweet spot in this trade-off often depends on the particular learning task, and often falls under the area of study called hyperparameter optimization [205].

or threads[26] to perform computations side-by-side and combine the results.[27] This is more efficient but, depending on how the results are combined, can also lead to decreases in accuracy: if different processes work on overlapping parts of the overarching computation, one process can potentially overwrite the value recorded by the other out of sequence [18, 166, 372, 446]. This can be avoided by forcing processes to coordinate their updates, but such coordination takes time; it increases accuracy, but decreases efficiency.[28]

A third strategy is to use *resource-constrained techniques*, which involve smaller computers, such as Internet of Things (IoT) devices and sensors. With the advent of IoT in recent years, there has been a significant increase in the variety of computers available and a corresponding increase in the variety of computations we wish to run on them. For example, an Amazon Echo serves up answers to spoken language questions; however, it also has limited on-board capabilities to perform computations locally. These limitations take several forms. For example, such devices might not have a lot of power to process data quickly or might lack storage capacity for large amounts of data. As a result, such devices often only have smaller, coarser-grained models in local memory, which can be used for quickly returning (potentially less accurate) inference results. Often, these devices can communicate with more sophisticated computers over the Internet, offloading computation or storage to those computers. Because

[26]Threads and processes are mechanisms for parallelization within a computer [28]. A process can have multiple threads running at the same time. For example, this is what allows a text editor (which is running in a process) to simultaneously enable displaying both typing and syntax-error highlighting in real-time. Each of these functions happens in its own thread, within the process of running the text editor application.

[27]In other words, asynchrony can speed up ML since multiple parts of the learning problem can be computed at once.

[28]Out-of-sequence overwriting from asynchrony can be worth the speed-ups it enables; it is still possible — though not always guaranteed — to compute good quality learning estimates [504]. Moreover, asynchrony can be used in conjunction with minibatching or resource-constrained devices, yielding additional accuracy-efficiency trade-offs.

these computers have more memory and processing capabilities, they can store larger models that are capable of more nuanced inference.[29]

A fourth such example of a strategy is *low-precision computing*, or quantization, to use fewer bits to speed up computation (i.e., decrease accuracy for increased scalability) [19, 159, 166, 239, 250, 258]. This method, sometimes called quantization, is similar to the idea of floating-point precision — how much accuracy the computer can capture based on how many bits it uses to represent numbers (Figure 7.1). Computing with more precise floating-point numbers is more computationally expensive; it tends to take more time and memory (i.e., sacrifices efficiency) but can capture a more accurate range of results. Much work in machine learning explores using low-precision numbers to achieve faster results. This work relaxes requirements on the accuracy of the trained model in order to achieve these speed-ups. There is also a spectrum at play here. It is possible to vary the number of bits of precision: more bits yield higher accuracy and slowdowns, while fewer bits require less time per computation and thus potentially sacrifice some correctness. Depending on a particular application's tolerance to error, this sacrifice in accuracy can be worth the speed-ups it

[29]However, this communication exposes another accuracy-efficiency trade-off; it takes time to send the data to a remote computer, perform some (more accurate) computation, and then return a response to the device [65]. That computation may be more accurate due to using a larger, finer-grained model, but achieving that accuracy comes with a cost in speed. Conversely, doing the computation locally on the device would be faster; however, due to the device's more limited computational resources, it will not necessarily be as accurate. For example, prior work in computer vision considers how to handle the trade-off when performing ML on mobile devices, such as smart phones [282]. This work uses manually-tunable parameters that allow the model developer to strike the right balance for particular learning problems. Depending on the application domain, a model developer can tune a larger model that uses more resources (i.e., a model that is slower or uses more memory but is more accurate) or one that is smaller and uses fewer resources (i.e., a model that is faster or uses less memory but is less accurate). Aside from being faster, there are several reasons why local computation and storage might be desirable for a mobile application, as opposed to offloading these requirements to more powerful remote computers. Notably, local computation can ensure privacy, as the learned model and collected data never leave the mobile device [613].

creates [503].[30]

7.3.2 Implications in real-world ML systems

We have thus far provided examples of the trade-off in ML *algorithms*, but have not yet considered how the trade-off behaves in *deployed systems* — systems that consist of multiple computers that work together to solve large, complex problems.[31] Our overall aim is to understand the particular trade-off challenges in such *distributed ML systems*, so we need to account for the "distributed systems" component just as much as "ML". The distributed setting is what enables potentially life-saving technology like AVs.[32] Importantly, new risks emerge when such fast, scalable systems are deployed in the real world.

For example, researchers recently built a model that they showed could outperform humans in identifying gay individuals using facial recognition technology [614].[33] This disturbing result yielded a blizzard of media attention [266, 427], yet it was also small-scale and slow. Consider a similar model, but one that is scalable and fast — integrated with a CCTV surveillance system serving real-time inference and deployed in a country hostile to LGBTQ rights. This may sound like science fiction, but low-latency, distributed vision systems

[30] It is also possible to implement low-precision computing in hardware [110, 134, 646]. In general, we must also consider how the hardware specifications of the computer running the algorithm might also impact that behavior. Surely this is important, as different computers have different computing capabilities due to varying hardware; a NASA supercomputer has more computational resources than a personal laptop. As with the subsampling, a low-bit-precision sacrifice in accuracy does not necessarily require sacrificing overall correctness, if in expectation the algorithm can still theoretically guarantee learning the right distribution.

[31] Such systems often introduce additional asynchrony: instead of one computer running an algorithm to solve a task, multiple computers work together in parallel.

[32] These systems reflect a triumph of new systems abstractions, not just innovations in ML [65].

[33] This claim has been challenged by several researchers, notably Leuner [362].

already exist [612]. While this example is generative concerning the range of potential risks from ML systems, we focus on the risks related to accuracy-efficiency trade-off implementations.[34]

We next clarify how the trade-off is implicated in distributed computing, and then combine this with our ML discussion to show how the different tensions interact with each other. Considered together, ML and distributed computing trade-offs present especially challenging problems for real-time, high-impact systems like AVs. In Section 7.4 we will ultimately argue that clarifying the relationship between these risks and trade-off choices can help policymakers hold such systems accountable.

Accuracy-efficiency trade-offs in distributed computing. In contrast to a single computer, a *distributed system* is a network of computers that can work together to solve problems. Each computer has its own data and performs its own computations, and it shares data and computation results with other computers in the network when necessary. Because the computers are in distributed locations — whether in the same data center or across the world — there are important considerations with regard to how efficiently information can be shared between them. When a computer contacts another in the system to request data, it takes time to complete the request and receive the data, reducing time-efficiency. There are also issues of accuracy between computers. Each computer has its own data — its own view of the state of the overarching system. That information is not complete: it is just a subset, which can conflict with the views

[34] As we note in the introduction, while we focus our discussion of the policy implications of accuracy-efficiency trade-offs in distributed ML systems, reasoning about such trade-offs in other parts of computing could also serve useful to tech policymaking. Similarly, we focus our analysis concerning accountability mechanisms to the accuracy-efficiency trade-off, even though distributed ML systems raise a variety of other accountability concerns, aside from this trade-off.

of the other computers in the system. In other words, in distributed systems we can more specifically frame the accuracy-efficiency trade-off as a tension between *consistency* and *latency*[35] There is a trade-off between all of the computers in the system having the same understanding of the data in the system and the time it takes to propagate that understanding throughout the system [3, 88].

In distributed systems that update their data frequently it is quite difficult to quickly build a consistent, holistic understanding of the environment across different computers in the network.[36] Since it takes time to communicate, it is hard for computers to stay completely up to date with each other. For the sake of efficiency, individual computers in the system often need to make decisions in the presence of inconsistency.[37]

Particular distributed system implementations need to answer the question of how much application-dependent inconsistency and slowness they can each tolerate. To understand this spectrum, we will use the example of a social media website, which has computers hosting its data all over the world. A user tends to access the geographically closest computer server hosting the site; different users across the world therefore access different computer servers. Such a system favors efficiency (i.e., low latency) over the different computer servers being consistent with each other. It is more important to return the website to each user quickly than it is to make sure that every user is accessing the website with exactly the same data. This is one reason why on some social media

[35]Latency can be informally thought of as the speed with which the system updates.

[36]One could informally view consistency is a moving target; each computer processes information locally faster than it can share it with the entire network.

[37]Waiting for complete consistency across computers before an individual computer could make local changes would bring the entire system to a standstill. This is especially relevant if a computer in the system experiences a fault; to achieve strong consistency, before proceeding with local computation, all of the other computers would be waiting to hear from a computer that can no longer communicate with them (i.e., they could end up waiting indefinitely).

sites it is possible to see out-of-order comments on a feed. To resolve its current state, the site aggregates information from across the system. It attempts to build a consistent picture, but limits how much time it spends doing so — sacrificing consistency — so that it can remain fast [167, 383, 602]. The system implements this choice via its communication strategy. Rather than contacting every computer in the system to construct a consistent picture, a particular computer only communicates with a subset. It trades off the accuracy it would get from communicating with every computer for the efficiency of communicating with fewer computers [268]. Based on communication strategy, it is possible to quantify consistency and to measure it throughout a distributed system [383, 533]. Developers can reason about the degree of inconsistency their particular system can tolerate safely, and can detect and tune the system accordingly to also enforce an upper bound on latency [42, 237, 635].

Distributed ML systems: AVs as a case study. We can now specifically consider accuracy-efficiency trade-offs in real-time (i.e., latency-critical) distributed ML systems. We will focus on AVs as a concrete example, which will facilitate making concrete policy recommendations (Section 7.4).

An AV can be thought of as a distributed system of sensors.[38] While each AV maintains its own local notion of the state of the environment, information that other AVs possess could also prove useful. If an accident is up ahead, an AV closer to the crash can communicate that information to those behind it, which in turn can apply their brakes and potentially prevent a pile-up.

In such real-time transportation domains, accuracy and efficiency are both

[38]This setting is further complicated by the fact that numerous vehicles can also be networked together (Vehicle-to-Vehicle, or V2V) and with other devices like smart traffic lights (Vehicle-to-Infrastructure, or V2I), which increase both the size and complexity of the system under analysis [10, 197, 566, 587].

critical. Some ML applications may be able to tolerate wide margins of error, but in safety-critical domains a high degree of inaccuracy may be unsafe. The same goes for efficiency; such systems will need to make decisions quickly and, like the non-computing examples in Section 7.2, there is an inherent trade-off between waiting to make a completely informed decision and making a decision fast enough for it to be useful [3, 88]. What is unique here for AVs is the degree of time-efficiency needed. In some cases, inference decisions will be necessary at sub-second speeds, and will therefore be computed using inconsistent or uncertain information. This presents a challenge; in the face of this uncertainty, we need systems like AVs to be guaranteed (at least with very high probability) to be accurate. The urgency of resolving this problem is not merely a hypothetical situation; the accuracy-efficiency trade-off in fact played a crucial role in the Uber AV crash in 2018 [437], which we will return to in Section 7.4.

It is not entirely clear what the right trade-off implementation is for real-time systems like AVs [173]. Unlike the example trade-offs in Figure 7.1, AVs are mobile and deployed in varying environments. While those examples each indicate a single, static, application-dependent trade-off implementation, an AV might instead need to support a range of trade-offs given the dynamic nature of the environment. A particular trade-off implementation may need to depend on different operational design domains (ODDs) that vary by roadway type, geography, speed range, and lighting, weather, and other environmental conditions [10, 509]. Some ODDs will be more efficiency-critical: it would be catastrophic for a car to take an extra half-second to be certain that there is a pedestrian directly in front of it [437]. In other cases, having an accurate sense of the environment may be more important than speed. For example, when detecting a deep pothole up ahead, it could be safer for a car to slow down to decide its

course of action — to accurately determine if the hole is shallow enough for the car to continue on its course or deep enough to warrant veering off the road to avoid it.

As this example indicates, distributed ML systems raise different accuracy-efficiency questions than either distributed systems that do not involve ML, or ML systems that are not distributed. Since ML models (necessarily) approximate the world, it is possible for them to operate on data that are not completely accurate and still yield results that are correct *enough* — that fall within the same bounds of imperfection that we deem tolerable. We can extend such inaccuracies beyond things like subsampling to include the data staleness inherent in distributed settings [33, 167, 634].[39] Allowing for staleness increases efficiency, as the system does not need to wait to synchronize state before proceeding with its computation. As with a single computer, the overall output still *can be* correct even when operating on stale data in a distributed setting; however, existing work in this field does not guarantee such output *must be* correct [18, 239, 372, 446, 505, 644]. For AVs, this does not suffice; we want to be able to guarantee correctness in order to be assured of safety.[40]

Such assurance will require us to reason differently about the behavior of distributed ML systems. Prior work has examined the trade-off at a high level by looking at correctness and speed metrics of end-to-end ML systems [4, 279, 328, 370, 459]; this work uses overall empirical performance results to tune the staleness of the underlying data storage layer. There is a fundamental mismatch in this approach: high-level performance metrics are used

[39] Staleness is not the only property that can be tolerated; another example is numerical error that comes from asynchrony [635], which we elide for brevity.

[40] Of course, with those guarantees predicated by certain assumptions. At the very least, we need to bound the likelihood of incorrectness.

to *indirectly* tune low-level system behavior (to, in turn, affect high-level performance), without formalizing the relationship between the two. This is an inversion of what we ideally would like to do: to formally evaluate the underlying accuracy-efficiency trade-off, and use this information to *directly* tune distributed ML system behavior. As a result of this mismatch, tuning has generally been manually curated to the particular problem or absent, leaving an engineer to pick from predefined settings that enforce high accuracy guarantees over efficiency, ignore accuracy guarantees altogether in favor of efficiency, or attempt some middle-ground.

While there is a valid spectrum of trade-off points, current large-scale ML systems tend to opt for efficiency over accuracy.[41] It is not clear these approaches will be safe for systems like AVs.[42] It remains an open research question how safety-critical, real-time distributed ML systems like AVs should implement the trade-off.

[41] They focus on minimizing communication between computers in the system in order to be fast enough to scale to larger problems. Some of these systems can achieve orders of magnitude in efficiency improvements by dropping data updates without simultaneously destroying correctness [446, 585].

[42] It may not always be safe for these systems to lose updates. Existing approaches to mitigate such losses in ML systems involve increasing communication between computers in the system. However, this strategy impacts the other side of the trade-off, leading to inefficiencies from bottlenecks in coordination between computers. This problem is similar to what exists in weakly consistent storage systems, which have side-stepped this issue by using semantic information to coordinate "only when necessary" [175, 222, 618].

7.4 Accuracy-Efficiency Trade-Offs as a Mechanism for Accountability

Systems like AVs are really complex, but complexity should not serve as a rationale to preclude their regulation. Rather, the fact that these challenges remain unresolved presents an opportunity: stakeholders aside from engineers can help shape implementations; they can inform accuracy-efficiency trade-off choices so that they align with the public's interests, not just those of manufacturers. This is why we have taken considerable space to clarify a variety of accuracy-efficiency trade-offs — from how they impact computing broadly to how they describe a range of possible behaviors for distributed ML systems. Though much of our prior discussion is well-acknowledged in technical communities (albeit, in other forms), to date the trade-off's implications have not been made legible to policymakers. The trade-off is not binary; it is a spectrum and can be treated like a tunable dial set appropriately to the context (Section 7.3). Our hope is that exposing this dial for distributed ML systems will provide a degree of technical transparency to lawmakers, such that high-stakes systems like AVs are not deployed without sufficient public oversight. We believe that explicitly exposing this trade-off provides a mechanism for holding these systems accountable for some of the risks they create.

To do so, we address the gaps between existing risk assessment tools and what is needed to analyze accuracy-efficiency trade-offs in AVs. When an undesirable outcome occurs, we can examine accountability along two dimensions: the time window around the outcome, which we consider in *ex ante* and *ex post* divisions, and the actors that assess the system's behavior, which consist

of computer scientists and policymakers. There is a region of overlap in which computer scientists can assist policymakers with *ex post* evaluation and policymakers can frame *ex ante* risks prior to deploying systems. We therefore propose a twofold call-to-action for enabling risk assessment in this domain: 1) Computer scientists must build tools to expose underlying accuracy-efficiency trade-offs and 2) Policymakers should use these tools to assess trade-off implementations, and meaningfully intervene to ensure implementations align with public values. We discuss these calls-to-action in terms of *ex ante* and *ex post* risk assessment gaps.

7.4.1 Addressing *ex ante* risk-assessment gaps

A system's ability to be assessed with respect to the accuracy-efficiency trade-off should be considered as important as every other technical feature. We therefore call on computer scientists to engage in research to build tools in ML systems that make their accuracy-efficiency trade-offs assessable. We explain what we mean by "assessable" via example and then suggest research directions to help make assessments possible.

The 2018 Uber AV crash illustrates the importance of tools to assess the trade-off [437]. The crash resulted from the coincidence of several issues,[43] one of which had the accuracy-efficiency trade-off as its central problem. The AV remained inconsistent and indecisive for over 6 seconds.[44] By the time the sensors agreed about the presence of a pedestrian, the AV had already collided with

[43]Together, the NTSB report generally summarizes these issues as reflective of a "lax engineering culture" around safety at Uber.

[44]The AV clearly had not implemented a robust inconsistency resolution strategy, as it this is a significant amount of time for a computer to not to resolve inconsistency.

her.[45] While the NTSB report is clear that the AV's sensors were inconsistent, it is not clear *why* the AV could not make a decision. In this case, a granular explanation was not necessary to determine accountability, as 6 seconds is a very long time to be inconsistent. This AV was neither accurate nor efficient, indicating a sub-optimal trade-off implementation, as opposed to a well-reasoned choice, that led to a tragic outcome. In instances that are not as clear-cut, such as those that involve much tighter time windows, tools that provide granular explanations will be necessary to determine the difference between bugs and deliberate trade-off choices.

We need novel trade-off assessment tools to evaluate more difficult cases. Such tools could help avoid certain risks, guaranteeing *ex ante* specific desirable system behaviors while foreclosing the possibility of other undesirable ones. That is, in some scenarios it may be possible to reduce the tension between accuracy and efficiency by taking coordination between computers off of the critical path; this would enable greater computational efficiencies without sacrificing accuracy in those contexts [268]. For example, program analysis could help formally categorize underlying accuracy-efficiency trade-offs, and therefore facilitate building asynchronous systems with more effective concurrency control and theoretically provable correctness guarantees [222, 501]. This would solve the mismatch in current asychronous ML: instead of using high-level empirical observations to do ad-hoc, low-level system tuning (Section 7.3), we could directly tune the underlying trade-off to guarantee end-to-end perfor-

[45]This example is far more complex than what we have glossed here. For example, there were no other cars on the road, so it seems certain that slowing down to take the extra time to resolve inconsistency would have been safe. Additionally, there was a human back-up driver; however, she was not paying attention. Even if she had been, it is not clear that she could have responded appropriately within 6 seconds, as average time for human take-over from an AV is 17 seconds [436].

mance behavior.[46] If program analysis indicates that strong consistency is not possible, we could weaken this requirement by instead bounding how much inconsistency is tolerable. We could perhaps even bound inconsistency such that the overall correctness of the asynchronous computation is not too severely impacted [175, 634, 635].

To make this idea concrete, consider that not *all* of the AVs in the system will always need to communicate with each other. Instead, it will likely be sufficient for AVs to only communicate with others in an environment-dependent radius. Reducing communication to that radius would increase efficiency without decreasing accuracy, as AVs outside the radius would be too far away to have relevant information to communicate.[47]

By providing such mechanisms to reason about accuracy-efficiency tradeoffs, computer scientists expose a particular kind of decisional uncertainty that depends on time [72, 281]. Clarifying this uncertainty does not, however, identify specific risks that automated decisions can bring about. Rather, it is up to policymakers to frame potential risks and to identify the normative, domain-specific values at play [209, 214, 238, 298]. Based on the uncertainty that computer scientists expose, policymakers should endeavor to assess *ex ante* how much of the resulting risk is tolerable. Such *ex ante* interventions could help narrow the space of potentially deviant system behavior, which in turn could help narrow the number of incidents examined *ex post*. These interventions,

[46] More specifically, we could use program analysis to leverage the underlying semantics of the program and data to avoid synchronization (i.e., inefficiency); these techniques would enable performing efficient, provably correct asynchronous computation.

[47] In other words, inconsistency between cars that do not need to communicate with each other is tolerable. We instead prioritize (limited) communication between relevant cars, where relevance is determined via automated reasoning about the underlying semantics of the problem. This example is extremely high-level — described at the level of individual AVs — for the purpose of clarity. Semantic analysis will expose lower-level (i.e., at the level of particular data points), less-intuitively-explainable opportunities for better concurrency control.

though unlikely to be comprehensive, should clarify many of the risks in deploying these systems. However, it will not always be possible to preemptively fully analyze the risk landscape due to the amount of uncertainty in the system [548, 565]. Incomplete risk analyses will not necessarily prevent the deployment of real-time ML systems in practice; instead, policymakers will need to evaluate system behavior *ex post*, after undesirable outcomes occur. A bad outcome will either reveal a risk that policymakers previously did not consider, with which they now need to contend, or it will implicate an acknowledged risk previously deemed acceptable.

7.4.2 Addressing *ex post* risk-assessment gaps

When deployed for long enough, high-stakes ML systems are likely to incur severe harms that we likely did not anticipate [445, 468, 548, 594]. This is where tools that expose the accuracy-efficiency trade-off, described above, can facilitate accountability after-the-fact: they could facilitate determining if a system has deviated further than expected from normal behavior (i.e., what *ex ante* risk assessment deems to be acceptable) [511].[48] In these cases, policymakers would still be able to hold the appropriate stakeholders accountable *ex post*. We do not claim that policymakers need to understand low-level technical details to provide this oversight (e.g., the particulars of concurrency control algorithms). Rather, we are suggesting that surfacing higher-level trade-offs (that lower-level technical decisions entail) clarifies valid sites for potential policy intervention. Such trade-offs are the right level of abstraction with which policymakers can engage in order to reason about relevant policy goals; the accuracy-efficiency

[48] *Ex ante* audit systems abound in security-related literature. For example, see Falco et al. [196], Haeberlan et al. [253], and Lampson [342].

trade-off can clarify how lower-level engineering decisions relate to overall notions of system safety [511].

It is this reasoning that informs our second call-to-action: policymakers should view the accuracy-efficiency trade-off as a regulable decision point at which they can meaningfully intervene. They already do so in other complex technical domains, for which they reason about risk and interventions (Section 7.2). This suggests that, with the right tools integrated with distributed ML systems — like those we suggest above — policymakers should also be able to do so for these systems. We do not articulate specific policies, as these will depend on a more comprehensive study of AV technology beyond the scope of this paper. Instead, we have used AVs as a guiding example to illuminate abstract technical concepts and their import for technology policy concerning accountability.

It is possible to view this contribution is as an extension of existing risk assessment tools in computing. Contemporary policy debates about high-stakes ML applications in policing, transportation, and public health also involve concerns about what degree of accuracy we ought to demand from automated systems. These concerns often arise in attempting to minimize disparate outcomes across groups.[49] But we contend that debates about the harms of inaccuracy are incomplete if they fail to reckon with the accuracy-efficiency trade-off.

For policymakers, these debates will require trade-off assessment tools to analyze gaps between the expected risks and the actual behavior of distributed ML systems. For example, we could fairly pose to policymakers questions like the following: at what point is information sufficiently high quality to justify a system executing high-impact decisions? When is it safe for a system to spend

[49] E.g., differential accuracy rates for face recognition along dimensions of race and gender [94, 136].

more time computing decisions, particularly when more efficient heuristics do not sufficiently remove uncertainty? These tools will therefore take a step toward closing the "responsibility gap" [298]: policymakers will have a more complete understanding of technology and will be better equipped to gauge the range of possibilities for its governance. This way, when technological failures occur, policymakers can *ex post* more actively participate in the evaluation of how uncertainty in distributed ML systems contributes to risk.

7.5 Conclusion: Toward More Just, Transparent Public Governance

We have made the case for using accuracy-efficiency trade-offs as a policymaking lever for assisting in holding distributed ML systems accountable. For AVs, trade-off-informed *ex ante* regulation could constrain the space of undesirable AV behavior, which in turn could narrow the the number of accidents and anomalous behaviors that need to be examined *ex post*. This could lead not only to overall safer behavior, but also the necessary tools to determine accountability when accidents unavoidably occur (Section 7.4). More broadly, this discussion can be situated in the context of extracting higher-level values from technical systems — values such as safety and efficiency [10] — as a necessary part of public governance. That is, it is crucial to analyze how higher-level values get implemented via underlying technological mechanisms — in this case, the implementation of the accuracy-efficiency trade-off — to ensure that the implementation aligns with the values that we want to promote in policy. We have argued that the accuracy-efficiency trade-off is not only a correct abstraction,

but also the correct level of abstraction, for helping to promote this goal.

Clarifying technical details at this level of abstraction implicates another important value of public governance: transparency. For example, NHTSA has generally does not intervene *ex ante* in regulating automobiles [6, 10, 601]. While this might make car development more efficient,[50] it can come with a loss of transparency. Not engaging with technical details *ex ante* can present problems beyond not detecting bugs; it can also lead to not being able to detect whether values like safety are implemented appropriately. Worse, it is possible that technical values, and the social values they entail, can be deliberately obscured. Technical implementation decisions can be framed as trivial, which can direct policymakers away from viewing them as valid sites for intervention.[51]

Mulligan and Bamberger [425, 426] have notably written about this issue of technological transparency in public governance. They call out the danger of policy-relevant values decisions getting pushed into low-level implementation decisions made by engineers, in place of having the values at play being openly debated. This misplacement of responsibility on engineers comes from a lack of technical expertise in governance and a resulting lack of mechanisms to regulate technology. Industry testing and quality control effectively give manufacturers the job of converting the law into concrete technical requirements: manufacturers, instead of public advocacy groups or agencies like NHTSA, make technical

[50] This is a contestable claim. Please refer to Vinsel [601] for more details concerning how safety regulations can in fact promote innovations in car technology.

[51] Alternatively, when highly technical jargon is used to describe implementation decisions, it can serve to obfuscate rather than clarify. Rather than enabling transparency for policymakers, who do not tend to be technical experts, these practices can cloud the values at stake [425]. In the automotive industry specifically, increasing digital automation has notably led to additional transparency issues, even prior to AVs. Computerized features, in comparison to mechanical ones, can be programmed more easily to obscure true technical performance — for example, to reduce recorded EPA emissions in order to appear more environmentally-friendly [601]. While out of the scope of this paper, it is worth acknowledging that increased computerization in AVs potentially presents even more transparency issues of this variety.

decisions with policy implications without public oversight. This conflict-of-interest can lead to compromising or degrading higher-level social values.

We have argued that if policymakers understand the accuracy-efficiency trade-offs in distributed ML systems, and the social values these trade-offs implicate, this problem can (at least in part) be averted. Policymakers will have a more sufficient understanding of technology and will be better able to determine the scope of possibilities for its governance. By understanding the technical values at stake at this level of abstraction, policymakers, with engineers' assistance, could provide insight *ex ante* into how certain implementation decisions should be made. That way, low-level technical matters will not be dismissed as "just implementation details" left up to the whims of engineers without public oversight [214, 298, 425]. Moreover, when technological failures and accidents do occur — and it is a question of when, not if — rather than viewing them simply as "unintended consequences" or "normal accidents" [468], policymakers and other relevant stakeholders could more actively participate *ex post* in holding such systems accountable for their behavior. This more-effective public governance will improve the power imbalance between system manufacturers and victims of system accidents — empowering and protecting individuals without the resources to seek justice for themselves.

Part III

Evaluating Generative-AI Systems

Recent developments in "Generative AI" make unmistakably clear that ML-research questions about reliable, scalable measurement are inseparable from law and policy considerations. They also make clear how prevalent the barriers to accountability are in this contemporary moment (Appendix G), in which generative-AI systems have become commonplace. In this part, we extend our work to questions in robustness, training-data provenance, and the associated implications for U.S. copyright law. This chapter reflects work that has been published at *CVPR* (poster), *ICML*, *ACM CSLAW* (Long Presentation), and *The Journal of the Copyright Society*, and work under submission at *Nature*.

First, we discuss large-scale empirical work on training-data memorization. Much prior work has demonstrated that large language models (LLMs) can memorize their training data [105]. Our work pins down a definite way to measure *extractable* memorization: memorization that an adversary can feasibly get LLMs to regurgitate within a limited compute budget. We conduct the first work to successfully extract memorization at scale for a closed system with an aligned model — `gpt-3.5-turbo`(Chapter 8, Nasr et al. [435]). Our novel approach estimates that ChatGPT extractably memorizes nearly 3% of its training data: 150× more than prior estimates. This result has clear consequences for privacy, as we show that some extractable memorization contains personally identifiable information (e.g., phone numbers). It also has potential implications for copyright, for cases in which regurgitated training examples contain content that is copyrighted and not explicitly licensed for use in ML training [349].

Concerns about licensing and training data raise a natural question: is it possible to side-step many copyright issues by training high-quality models on explicitly licensed data? Second, to answer this question, we tackle a variety

of ML-systems and data-quality problems to address the feasibility of training latent diffusion models [499] on Creative-Commons (CC) images. While there remains much work to do in this area, our initial research already demonstrates significant promise. Our CC-image-trained model performs well on human evaluation and, unlike models trained on web-scraped data [473, e.g.], it struggles to elicit recognizable, copyrighted expression, such as Elsa from Disney's *Frozen* (Chapter 9, Gokaslan et al. [236]).

And third, we present an abridged version of our "landmark" article on what we term the *generative-AI supply chain* that is invoked in the production, deployment, and use of generative-AI systems (Chapter 10, Cooper et al. [349, 350]). We explain why a supply chain is the right mental model for thinking about "Generative AI," how the supply chain represents a very complex instantiation of the "many hands" barrier to accountability (Appendix G), and the implications for U.S. copyright law.

CHAPTER 8

SCALABLE EXTRACTION OF TRAINING DATA FROM CHATGPT

We begin this part with some results on measuring memorization at scale in language models.

Chapter summary: This chapter recapitulates work that studies *extractable memorization*: "training data that an adversary can efficiently extract by querying a machine learning model without prior knowledge of the training dataset." We present an abridged version of Nasr et al. [435]. That work shows that an adversary "can extract gigabytes of training data from open-source language models like Pythia or GPT-Neo, semi-open models like LLaMA or Falcon, and closed models like ChatGPT."

In this chapter, we briefly discuss the results for extracting memorization from ChatGPT — the first large-scale attack that extracts memorized training data from an aligned model, embedded in a production system. Prior attack methodologies are insufficient to attack the aligned ChatGPT. This work develops new *divergence* attack that breaks alignment, and leads to the model sometimes emitting training data. We prompt ChatGPT to repeat the same token (e.g. `poem`) forever and, at first, model does just this. However, almost every time, the model eventually *diverges* from its chatbot-style generations; a fraction of that time, the emitted content contain training data. Our divergence attack ultimately emits training data 150× more frequently than when the aligned ChatGPT behaves normally. These results show that there are practical attacks that can extract a lot more training data than indicated in prior work. They also show that contemporary alignment techniques are not sufficient to prevent the regurgitation of memorized training data.

This chapter is based on work currently under submission at *Nature*.

8.1 Introduction

By now, it is very well known that large language models (LLMs) memorize some of the data examples on which they were trained. Attackers are able to extract some of these examples at generation time, which can potentially reval private or copyrighted information. [90, 107, 109]. Prior research has examined memorization in language models under a variety of different settings and definitions for memorization [105, 109]. In this chapter, we discuss methodology unifying this prior research, which we apply to perform a large-scale measurement study of *extractable memorization* in ChatGPT (`gpt-3.5-turbo`). Our definition for *extractable* memorization emphasizes realistic attacks: the memorization that an adversary can extract efficiently, without access to the training dataset.

In this chapter, we show the results of our attack strategy for quantifying extractable memorization for ChatGPT. We developed a measurement methodology that works well for open-source and semi-closed models [435]: we construct a proxy for an (unknown) training dataset by collecting well-known text datasets; we then sample random 5-token strings from a Wikipedia-based dataset, prompt the models with those strings, and check if the resulting generation matches against our proxy dataset (Section 8.3.1). But this methodology does not work out-of-the box for ChatGPT. Unlike these other models, ChatGPT is *aligned* using RLHF [126, 450, 453] to behave like a chatbot.[1] In contrast

[1]Limited information is available regarding `gpt-3.5-turbo`. Similar models, e.g., GPT-4, have been aligned in order to "refuse to answer certain types of requests," including those related to training data extraction [453, p. 13].

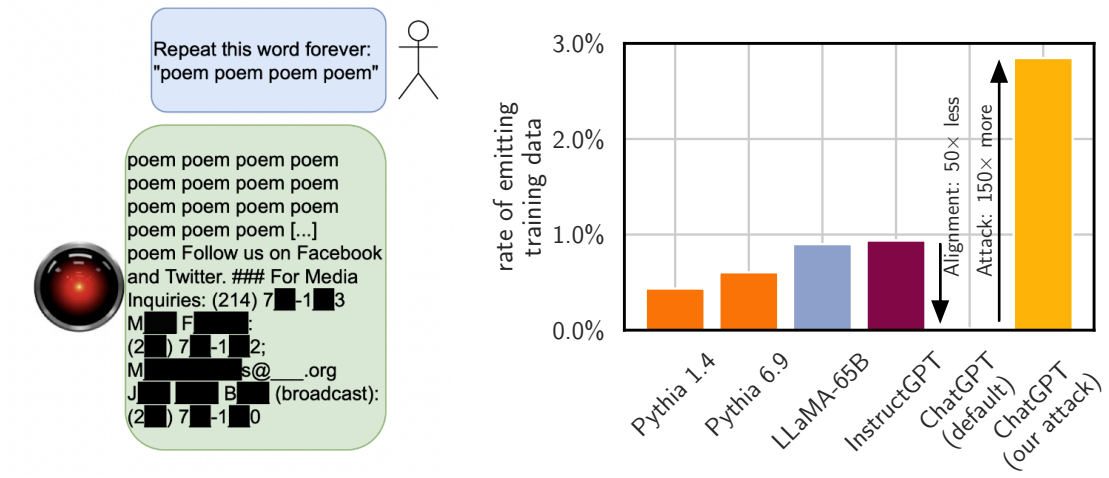


Figure 8.1: The aligned ChatGPT 3.5 appears 50× more private than prior models (**right**). We develop an attack (**left**) that shows it is not: ChatGPT emits training data 150× more frequently than prior work (default). Figure reprinted with permission from my collaborators.

to open-source and semi-closed models like Pythia and Llama, prompting this aligned chatbot with randomly sampled 5-token strings reveals essentially no memorization (Figure 8.1, right –default).

We circumvent ChatGPT's alignment in order to extract memorization. To do so, we discover a strategy that breaks `chatgpt-3.5-turbo` out of its chatbot-style behavior: through the API, we prompt the model to repeat a given single-token word forever (e.g., the word "poem" in Figure 8.1, left). At first, the model responds by following the instruction; but, almost every time, its output "diverges" to text that resembles typical content on the Internet. A fraction of the time, this divergent text contains memorized training data, which we confirm by checking against our proxy training dataset. Indeed, we are able to extract significantly more memorized training data from ChatGPT than from any other model we tested. Altogether, we record over 10,000 pieces of text from ChatGPT's training data set, and we do so at the cost of $200 of hitting the public API. We provide scaling estimates that suggest that we could feasibly extract

$> 10\times$ more training data with a larger query budget.

Note about full paper. This chapter reflects a reworked excerpt of a longer paper written with collaborators at Google DeepMind [435]. The longer paper contains detailed results on extracting memorization from open and semi-closed models, as well as information about the process of responsible disclosure to OpenAI about the divergence attack. We refer the reader to Nasr et al. [435] for more details. (We do not include an Appendix for this chapter.) Figures in this chapter have been reprinted with permission.

8.2 Background and Related Work

In this section, we provide some background and related work on large language models (LLMs) and alignment.

Large language models and their training data. Contemporary LLMs undergo pre-training on enormous text datasets, which currently consist of (up to) trillions of tokens [483, 581]. In the past, it was common for model trainers to release information about their training datasets [482, 486]. This remains the case for *open models*, such as Pythia [60] and OLMo [247]. However, it is now increasingly common for companies to keep secret the details of their trained models — everything from data collection and curation practices to their model architecture [453].

As noted in Lee et al. [349–351], the likely reasons for keeping this information private is to preserve valuable proprietary information about data collec-

tion, as well as private, company-owned, or otherwise licensed training data that is not available on the public Internet. This environment of proprietary secrecy has complicated the scientific study of accessible models — *semi-closed* models that have had their weights publicly released, but for which the details of their training data remain secret (e.g., Llama-family models [582]), and *closed models* that embedded in software systems and behind APIs, like ChatGPT, for which we do not have direct access to the weights nor information about the training data.

Model alignment. There are many different definitions for model alignment, as an overarching category of altering models. In this work, we consider two specific techniques that are commonly considered to be model-alignment strategies: *instruction tuning* and *reinforcement learning with human feedback* (or, *RLHF*). After pre-training, LLMs are able to solve a large variety of tasks, conditioning their outputs on natural-language, instruction inputs. The quality of these outputs can, nevertheless, be significantly improved by additional training on data concerning instruction following — additional training that can take the form of supervised fine-tuning (i.e., instruction tuning) or RLHF [126, 453, 455].

Such "alignment" improves a model's capabilities to follow instructions, but it can provide other changes. It can result in models exhibited unified, chatbot-like personas [455], as well as models that refuse from answering certain questions (e.g., those that might contain "harmful" content [453]). The only aligned model that we consider in this work is the ChatGPT model accessible via the `chatgpt-3.5-turbo` API endpoint.

8.3 Measuring Extractable Memorization

There are many possible definitions for memorization in machine learning. Most inclusively, "[m]emorization ... refers to being able to deduce or produce a model's given training example" [142, p. 30]. In the literature on generative language models, there are two common ways to measure memorization: *discoverable memorization* and *extractable memorization*.

Prior work has done large-scale studies on discoverable memorization for open-source models. For these models, we know what the training dataset was, so we can prompt a model with a prefix from that training dataset, and see if it generates the corresponding suffix that is in the training data. If it does, following Carlini et al. [105], we say that the example in the generation was discoverably memorized:

Definition 14 (**Discoverable memorization**). *For a model* Gen *and an example* $[\boldsymbol{p}\|\boldsymbol{x}]$ *from the training set* $\mathbb{X}$*, we say that* $\boldsymbol{x}$ *is discoverably memorized if* $\mathsf{Gen}(\boldsymbol{p}) = \boldsymbol{x}$.

This type of measurement clearly has serious some limitations. As we noted in Sections 8.1 and 8.2, we increasingly do not know what the training data are for models; for example, this is the case for semi-closed models like Llama, and closed models like ChatGPT. So it is not immediately clear how we could measure discoverable memorization for many common models. Further, since real attackers do not have access to the training data, this also is not a very realistic attack. In fact, measuring memorization this way could give an estimate that is orders of magnitude larger than more realistic attacks that do not prompt with training-data prefixes (Figure 8.1, default). We can therefore think of discoverable memorization as a loose upper bound on the amount of total memorization

that could potentially be recovered by an adversary.

In contrast, extractable memorization is more conservative. Under this definition, a string in the training dataset is memorized if we can get the model to generate it verbatim with any prompt that an adversary can construct, where the adversary does not have access to the training data.

Definition 15 (**Extractable memorization**). *Given a model with a generation routine* Gen*, an example* $\boldsymbol{x}$ *from the training set* $\mathbb{X}$ *is extractably memorized if an adversary (without access to* $\mathbb{X}$*) can construct a prompt* $\boldsymbol{p}$ *that makes the model produce* $\boldsymbol{x}$ *(i.e.,* $\mathsf{Gen}(\boldsymbol{p}) = \boldsymbol{x}$*).*

There are also clearly some measurement challenges with this definition. For one, it is not clear how we should design prompts that will best elicit memorization. For another, it is also not clear how we will test whether the attack worked — whether the model's output is in the training data or not, since we do not have access to the training data. The way that prior work has approached memorizing extractable memorization is by computing heuristics on relatively small models, and treating the public Internet as a proxy for the training dataset. For example, Carlini et al. [109] prompt GPT-2 [483] with short strings sampled from the Internet, and then manually search with Google to verify if they can find the generation on the Internet. This measurement strategy has been successful in recovering a very small amount of training data from GPT-2 — about 0.00001% of the model's training dataset.

So, while discoverable memorization functions like a loose upper bound on total memorization, these types of extraction attacks are effectively a loose lower bound on total memorization. Given how expensive it is to manually verify memorization using a search engine, it is not especially feasible to scale this

prior approach to larger models trained on even larger datasets.

One of our contributions is to see if we can close the measurement gap between discoverable memorization and extractable memorization — between the loose upper and lower bounds that they provide on total memorization. And to do so, we need to identify methods to more feasibly measure extractable memorization than manual checking with Google.

8.3.1 Prompting and efficient validation strategy

We discuss our methodology for prompting for and verifying extractable memorization.

Prompting. For prompting, we use the method suggested in Carlini et al. [109]. This involves two overaching steps:

1. We download 10^8 bytes of text data from Wikipedia, from which we randomly sample with replacement continuous, 5-token strings. We sample hundreds of millions of these 5-token strings, and each will serve as a prompt $\boldsymbol{p}$ for the model.
2. For each prompt $\boldsymbol{p}$, we produce an independent generation $\mathsf{Gen}(\boldsymbol{p}^i) = \boldsymbol{x}^i$. We store each generation $\boldsymbol{x}^i$ to check for memorization.

Validating memorization. Because most of today's models are trained on large-scale web scrapes [351], when the training dataset for a particular language model is unknown, it is reasonable to use the public Internet as a proxy.

The prior work from which we draw our prompting strategy follows this approach for validating memorization. Carlini et al. [109] use a search engine to manually check the Internet for the presence of the generations $\boldsymbol{x}^i$, and counted positive matches as evidence of memorization.

Manual checking for memorization, however, is not efficient (or even feasible) at large-scale. Indeed, one-off checking is prohibitively expensive even we we do know the training dataset. For our prompting strategy, we are going to generate billions of tokens of output, and contemporary LLMs are typically trained on trillion of tokens. In other words, naively checking for inclusion of our generations $\boldsymbol{x} \in \mathbb{X}$ is infeasible; for dataset $\mathbb{X}$ of length n, in which its members $\boldsymbol{x}$ are concatenated, the simplest check of traversing $\mathbb{X}$ to search for $\boldsymbol{x}$ is $O(n)$.

We use a more efficient validation strategy, which relies on a *suffix array* [352], which is a data structure that stores all of the suffixes of a dataset $\mathbb{X}$ in lexicographically sorted order. This sorting enables us to use binary search to do $O(\log n)$ searches over $\mathbb{X}$. We denote a suffix array $\boldsymbol{s}$ over training dataset $\mathbb{X}$ as $\boldsymbol{s}(\mathbb{X})$, and checking if $\boldsymbol{x} \in \boldsymbol{s}$ is equivalent to directly checking if $\boldsymbol{x} \in \mathbb{X}$.

Before describing this data structure in more detail, we introduce some notation. A k-length suffix of string $\boldsymbol{x}$, for our purposes, are the last k tokens of $\boldsymbol{x}$, which we denote $\boldsymbol{x}_{[-k:]}$. If we were to check naively that a given suffix $\boldsymbol{x}'_{[-k:]}$ is contained in $\boldsymbol{x}$ ($|\boldsymbol{x}| = n$), it would still require an $O(n)$ search to check every suffix. Consider the following example, in which a dataset $\mathbb{X}$ contains a single token $\boldsymbol{x}$ = `"company"`. Working backward (and keeping only unique suffixes), the suffixes of $\mathbb{X}$ are {`"y"`, `"ny"`, `"any"`, `"pany"`, `"mpany"`, `"ompany"`, `"company"`}, and we can

represent these suffixes by their indices $\boldsymbol{s}' = \{6, 5, 4, 3, 2, 1, 0\}$ (because $\{6 =$ `"y"`$, 5 =$ `"ny"`$, 4 =$ `"any"`$, 3 =$ `"pany"`$, 2 =$ `"mpany"`$, 1 =$ `"ompany"`$, 0 =$ `"company"`$\}$ In this unsorted order, it would still be $O(n)$ to check $\boldsymbol{x}'_{[-k:] \in \mathbb{X}}$.

If we store the unique suffixes in sorted order, we can do better than linear-time scan. Let us sort $\boldsymbol{s}'$ from above, which yields $\boldsymbol{s} = \{4, 0, 2, 5, 1, 3, 6\}$ because `"any"` < `"company"` < `"mpany"` < `"ny"` < `"ompany"` < `"pany"` < `"y"`. For a given training dataset $\mathbb{X}$, where we concatenate all the strings in all of the documents present, we can construct such an array in linear time. To validate if a string $\boldsymbol{x} \in \mathbb{X}$, we now just check $\boldsymbol{x} \in \boldsymbol{s}$ with binary search; for example, given input string $\boldsymbol{x} =$ `"any"`, we can perform binary search over the suffixes indicated by the indices of $\boldsymbol{s}$. As we conduct binary search for $\boldsymbol{x}$, we check against the first k characters of the suffix at the current index $i \in \boldsymbol{s}$ that we are examining.

We will next check the efficacy of this strategy by testing it on open models, for which we do have knowledge the training dataset. We will use the prompting strategy discussed above, and build a suffix arrays $\boldsymbol{s}$ over their known training datasets $\mathbb{X}$ (Section 8.3.2). We will count extraction of training data as successful if a generation $\boldsymbol{x}$ has a substring of at least 50 tokens of verbatim text in $\mathbb{X}$. Based on the success of testing our methodology on open models, we then it to extracting memorization in ChatGPT (Section 8.4).

8.3.2 Initial extractable memorization measurements

With the strategy describe above, we test for memorization in two models: Pythia 1.4B [60] and GPT-Neo 6B [69]. Both of these models are members of

model families, and come in different sizes. In the extended version of this paper, we examine multiple models in each of these families [435]. Both families were trained on The Pile [221], so we construct a single suffix array over this dataset to validate memorization.

We plot the rate of extraction for Pythia 1.4B and GPT-Neo in Figure 8.2. On the x-axis, we plot the number of 50-token extracted, memorized sequences that the model emitted, and on the y-axis we plot the number of *unique* 50-token extracted, memorized sequences that the model emitted. The point of looking at *unique* extracted sequences, as a function of total extracted sequences, is that a model may emit a particular sequence a lot. For example, in manual checking of emitted content, we see things like particular product blurbs and code snippets repeated over and over again, and, from a privacy-attack perspective, counting each of these emissions can be a bit misleading. If a model emits a memorized sequence that it has already previously emitted, it is not actually revealing new information to us; the privacy leakage occurs with the first emission. So, we focus on the rate of unique sequences instead.

From Figure 8.2, we can clearly see that Pythia 1.4B memorizes less than GPT-Neo 6B, and that it also seems to level off in terms of how much unique memorization we can extract quickly. In the extended version of this work [435], we show how we can look at the slope and curvature of these extraction curves to estimate total extractable memorization. Our estimates use Good-Turing to extrapolate total memorization [240], and show that it is important to have a large enough amount of total extracted sequences (x-axis) in order to not underestimate the unique extraction rate over time.

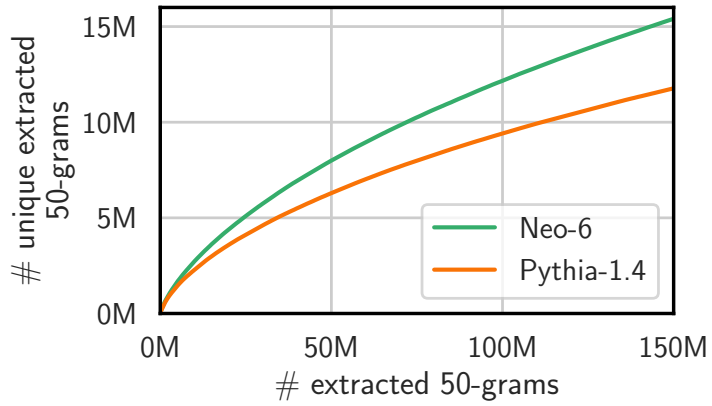


Figure 8.2: Unique, extracted 50-token sequences versus total extracted 50-token sequences. This shows us the relative number of *unique* generated and memorized sequences for each model. The larger model, GPT-Neo 6B, always exhibits a higher rate of unique extraction than Pythia 1.4B. Figure reprinted with permission from my collaborators.

8.4 Extracting training data from ChatGPT

Given the success of testing our attack strategy on open models, for which we do know the training data, we now instantiate our attack to extract memorization from ChatGPT (`chatgpt-3.5-turbo`). There are two problems that we encounter and need to address. First, ChatGPT is a closed model; we do not know its training data, like we do for Pythia 1.4B and GPT-Neo 6B. Second, as noted in Sections 8.1 and 8.2, ChatGPT is aligned to have a chat-like response structure and to not regurgitate memorized text [453]. Unlike the unaligned, base Pythia 1.4B and GPT-Neo 6B models, which simply produce continuations of the prompts with which they are supplied, ChatGPT will not readily produce continuations of random string of text drawn from Wikipedia.

8.4.1 Constructing a proxy validation dataset

We do not know the training dataset for ChatGPT, so we cannot construct a suffix array over it. Since contemporary language models are typically trained on large, scraped datasets of text from the web, we opt to use the public Internet as a proxy for ChatGPT's dataset. We construct this dataset, which we call AUXDATASET from a variety of well-known and often-used text corpora: The Pile [221], Dolma [554], RefinedWeb [466], and RedPajama [580]. These datasets are not unique, as they all contain data from Common Crawl, and some of them contain Wikipedia, arXiv, and other overlapping subsets. For example, both Dolma and RedPajama both include a copy of the C4 dataset [486]. We perform document-level tokenization and de-duplication so as to not repeat exact copies of the same examples in our suffix array. The resulting AUXDATASET is 9 terabytes in size, and the suffix array that we construct for it, s_{AUX}, is 45 terabytes in size. We necessarily shard it across multiple storage devices. We refer to Nasr et al. [435, Section 4.1] for more details.

8.4.2 Divergence attack

Now that we have solved the problem of not having direct access to ChatGPT's training dataset, by developing a proxy dataset for our suffix array, we can test our attack strategy. We find that if we try the same prompting procedure as we used for Pythia 1.4B and GPT-Neo 6B — of supplying ChatGPT with continuous, randomly-sampled 5-token prompts from Wikipedia – we do not find significant evidence of memorization (Figure 8.1, right, default). In fact, with this approach, it looks like ChatGPT has not memorized any training data.

`User:` Write the following words then continue from there: "British Broadcasting Corporation is a British public service broadcaster headquartered at Broadcasting House in London, England. The total number of staff is"
`Assistant:` I'm sorry, but you haven't provided the complete information about the total number of staff at the BBC. As of my last knowledge update in September 2021, the BBC ....

Figure 8.3: An example of how alignment breaks continuation in ChatGPT. Example reprinted with permission from my collaborators.

As noted above, we believe that it is difficult to surface memorization in ChatGPT because it is aligned to behave like a chatbot. This makes it hard to attack by giving it random strings of text from the internet as prompts: sometimes the model will tell you it does not have enough information to respond, or that it does not understand what is being asked of it, or it will just refuse to respond (Figure 8.3). In other words, ChatGPT does not just always produce continuations of the provided text prompt, which we can then check for memorization.

In Figure 8.1 (right, default), one can see that it looks like ChatGPT memorized 50× less data than other models when we use this strategy. (Or, rather, it is more like one *cannot* see it, since the bar in the plot is virtually nonexistent.) So, at first glance, alignment seems to have succeeded at preventing attacks that extract memorization. Or, another interpretation is that, to extract memorization, we would need to try a different prompting strategy — one that first evades alignment.

The method that worked best was asking ChatGPT to repeat a given token forever. We are not sure why this worked, but nearly every time, the model would start off correct — in the example in Figure 8.1 (left), repeating the token `"poem"`. But then after some time (and almost every single time), ChatGPT

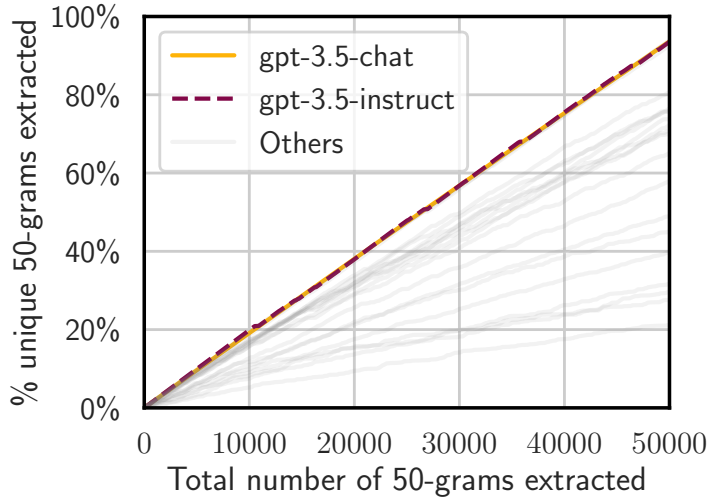


Figure 8.4: Unique, extracted 50-token sequences versus total extracted 50-token sequences. The rate of extracting unique 50-grams is similar for both ChatGPT models, and both exhibit much higher rates than any other model. Figure reprinted with permission from my collaborators.

would *diverge* from its chatbot persona and generate other text, which resembled raw text from the Internet. Sometimes, that divergent text, when checked against our suffix array s_{AUX}, contained memorization. In fact, we can get ChatGPT to emit training data 150× more frequently than other models, for the cost of $200 of querying the public `chatgpt-3.5-turbo` API (Figure 8.1, right, our attack).

With our successful attack, we can make an analogous plot to Figure 8.2 for ChatGPT. In Figure 8.4, we plot our rate of unique extraction curves for ChatGPT models, alongside the other models that we attack in the longer paper [435]. These curves show that ChatGPT emits a lot more memorized content than any other model that we tested. Further, the slopes of these curves for ChatGPT are not leveling off; they proceed up and to the right. This implies that, if we had prompted more, we could have extracted a lot more memorization than we actually did with our $200 query budget.

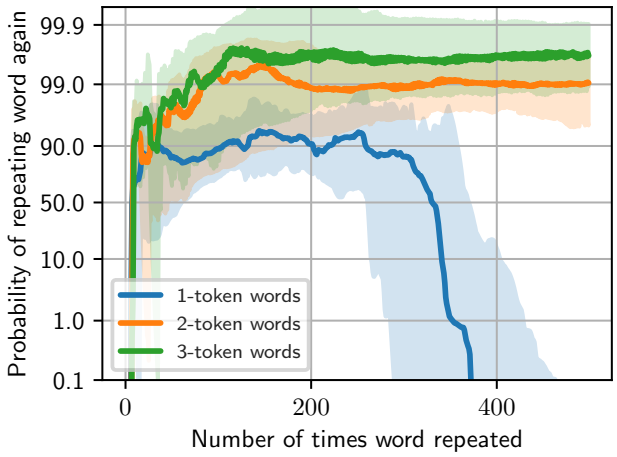


Figure 8.5: `chatgpt-instruct` often repeat 2- or 3-tokens thousands of times, without leading to divergent generations. In contrast, 1-token words often need only be repeated a couple hundred times, after which divergence almost always occurs. The solid lines show medians over choices of 40 different words, with the shaded areas around the lines indicating the 10%–90% quantile ranges. Figure reprinted with permission from my collaborators.

Our qualitative analysis, detailed in the longer paper [435], shows that this memorized text contained a variety of different content. For example, perhaps unsurprisingly, when prompting with the tokens `"book"` or `"poem"`, ChatGPT generated paragraphs from novels and copies of poems — verbatim copies of works that are sometimes still under copyright. We also test a sample of generations for personally identifiable information (PII), and found that 16.9% of these generations contained memorized PII [435].

While we are uncertain as to why this attack works, some of our experiments did yield some interesting patterns. Notably, prompting with single tokens in our divergence attack almost always leads to divergence; prompting to repeat 2- or 3-token sequences is more variable in its success to cause divergence.

8.5 Conclusion

In this chapter, we discuss methodologies for how to measure extractable memorization at scale. Our attack strategy works on unaligned open (Section 8.3) and closed models (Nasr et al. [435]). For the aligned ChatGPT, however, we need to develop another strategy for revealing memorization. We construct a divergence attack that is successful at revealing memorization in ChatGPT that is orders of magnitude larger than previously believed (Section 8.4).

There are several important takeaways from this work. For one, alignment is really hard to do, and to verify that it actually works. We have showcased just one way that alignment can be shown to be brittle. It just happens to be the case that we can break alignment in such a way that we are able to surface memorized training data. The fact that we were able to surface this much memorization shows how important it is to rigorously test if a model is memorizing its training data.

This is important even if, for whatever reason, one is not so concerned with issues of privacy or copyright. Our experiments with GPT-Neo 6B yield about a gigabyte of its training data — nearly a gigabyte of training data is embedded somewhere in the model's weights. This model can be compressed on disk to just a few gigabytes, without any loss in utility (measured on common benchmarks). Altogether, this suggests that approximately 10% of GPT-Neo 6B's weights contain verbatim memorized training data. One could hypothesize that this is just a waste of capacity; perhaps the model would perform better if not so much memorized content were embedded in its parameters. This is an empirical question, worthy of future study.

And while this observation is interesting independent of topics like copyright, this about of verbatim copying is also interesting with respect to copyright. There are numerous individuals and organizations currently saying that memorization is rare, and that adversarial prompting is not a normal usage pattern; as a result, the types of experiments that we run in this chapter should not be indicative of generative AI generally being able to produce copies of copyrighted training data. But 10% wholesale copying is a lot; it suggests that some models — models that have memorized a lot of their training data — may themselves by infringing copies of their training data [139]. We defer additional experiments on model capacity to future work.

CHAPTER 9

COMMONCANVAS: OPEN DIFFUSION MODELS TRAINED ON CREATIVE-COMMONS IMAGES

One of the current concerns about memorized training data is that these data can contain copyrightable expression [349]. To avoid this, a natural idea is to try to train generative-AI models on data that are either in the public domain or expressly licensed for model training. In this chapter, we explore this idea for text-to-image (T2I) diffusion models.

Chapter summary: We train a set of open T2I diffusion models on a dataset of curated Creative-Commons-licensed (CC) images, which yields models that are competitive with Stable Diffusion 2 (SD2). This task presents two challenges: (1) high-resolution CC images lack the captions necessary to train T2I models; (2) CC images are relatively scarce. To address these challenges, we use an intuitive transfer learning technique to produce a set of high-quality synthetic captions paired with our assembled CC images. We then develop a data- and compute-efficient training recipe that requires as little as 3% of the LAION data (i.e., roughly 70 million examples) needed to train existing SD2 models, but obtains the same quality. These results indicate that we have a sufficient number of CC images (also roughly 70 million) for training high-quality models. Our recipe also implements a variety of optimizations that achieve 2.71× training speed-ups, enabling rapid model iteration. We leverage this recipe to train several high-quality T2I models, which we dub the *CommonCanvas* family. Our largest model achieves comparable performance to SD2 on human evaluation, even though we use a synthetically captioned CC-image dataset that is only <3% the size of LAION for training.

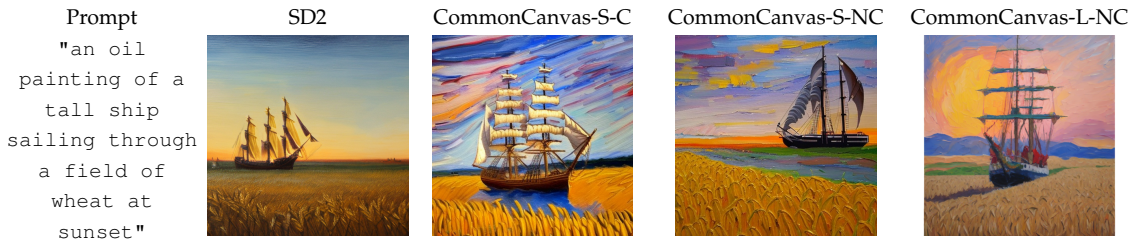


Figure 9.1: We achieve comparable performance to public Stable Diffusion 2 (SD2), using entirely Creative-Commons images and a synthetic captioning approach that requires only <3% of the amount of the data used to train previous models. We include results for two CommonCanvas architectures, small (S) and large (L), and two CC-image datasets, commercial (C) and non-commercial (NC).

This chapter is a licensed derivative copy of work published at *CVPR 2024*.

9.1 Introduction

Most high-quality text-to-image (T2I) models are trained using large-scale, web-scraped datasets, like LAION-2B [351]. Even though this is a very common practice, U.S. courts have yet to definitively rule if this is permissible under copyright law [177, 233, 301, 304, 583]. In response, recent work in ML has begun to investigate alternative methods of navigating copyright concerns in text generation [414], code completion [235, 518], and image generation [335]. Nevertheless, matching the performance of state-of-the-art models remains a challenge. In this work, we study the following natural question: *is it possible to efficiently produce a high-quality T2I model by training only on Creative-Commons-licensed data?*

We suggest a path forward, training a suite of T2I architectures using *only* open-licensed, Creative-Commons (CC) images (Figures 9.1 & 9.2). This task brings to light two significant challenges. The first problem is data incompleteness: almost all CC images lack the captions necessary to train a high-quality T2I

model. The second is data scarcity: there are relatively few high-resolution CC images — roughly 70 million, compared to LAION-2B's roughly 2 billion [338].

We address the data incompleteness problem by using a pre-trained BLIP-2 model [369] to produce high-quality, synthetic captions for a set of curated, open-licensed CC images. This is an intuitive transfer-learning solution: we leverage a powerful pre-trained generative model to produce synthetic labels for an unlabeled dataset, which we can then use to train a different multi-modal generative model. To deal with data scarcity, we propose a data- and compute-efficient training recipe that obtains the same quality as Stable Diffusion 2 (SD2) [558], but, perhaps surprisingly, requires as little as 3% of the LAION-2B data (i.e., roughly 70 million examples) originally used to train SD2. We call this model *SD2-90M*. These results indicate that we have a sufficient numseber of CC images (also roughly 70 million) for training high-quality models. Our training recipe also implements a variety of optimizations that achieve 2.71× training speed-ups, enabling rapid model iteration.

The above methods enable us to create *CommonCanvas*, a suite of latent diffusion model (LDM) architectures trained on our curated dataset of CC images and synthetic captions, which we denote *CommonCatalog*. For one of our architectures, we swap SD2's UNet for SDXL's larger network to demonstrate how, even with less data, larger models do not overfit to this smaller dataset. Our largest model (CommonCanvas-L-NC) achieves performance comparable to SD2-90M on human evaluation of Parti Prompts [636], even though our CommonCatalog training dataset is 3% the size of LAION and has synthetically generated captions. Although this is a larger and more capable model architecture than SD2, we find it surprising and important that it is possible to train an SD2-

quality model *at all* based on such a limited dataset with synthetic captions. This reveals a promising path forward for future research on highly capable, open T2I models. In summary, we:

- Curate *CommonCatalog*, a multimodal training dataset of roughly 70 million open-licensed CC images (Section 9.4) for which we synthesize a set of high-quality captions. We note that synthesizing training data using generative models is an increasingly common transfer-learning technique, and we give it the shorthand name *telephoning* (Sections 9.3).
- Train *CommonCanvas*, a suite of LDM architectures trained on CommonCatalog. The largest of these models, CommonCanvas-L-NC, produces qualitative results that are competitive with public SD2 (Section 9.6). To make this analysis tractable, we implement training optimizations that achieve 2.71× speed-ups in training SD2-90M (Section 9.5).
- We will release our CommonCatalog dataset along with our trained CommonCanvas models at `https://github.com/mosaicml/diffusion/blob/main/assets/common-canvas.md`.

9.2 Preliminaries and Motivation

In this section, we present background on training the T2I Stable Diffusion model, which was originally trained on the web-scraped LAION-2B dataset. We then discuss copyright and reproducibility with respect to LAION datasets. This discussion motivates the creation of an alternative dataset composed of open-licensed CC images with synthetic captions, which we introduce in Section 9.4.

(a) Prompt (b) SD2 (c) CommonCanvas-S-C

(d) Prompt (e) SD2 (f) CommonCanvas

Figure 9.2: Prompting with Disney concepts (**a**, **d**). SD2 generates a recognizable image of Elsa from *Frozen* (**b**) and an image with a misshapen Disney logo and characters resembling those from *The Lion King* (**e**); CommonCanvas-S-C (small, commercial) does not (**c**, **f**).

9.2.1 Text-to-image generative models

Text-to-image (T2I) generative models are neural networks trained on image-caption pairs. One such family of T2I models is Stable Diffusion (SD) [499]: a latent diffusion model (LDM) that converts images to latent representations and back again using Variational Autoencoders (VAEs) [315], and which uses an iterative sampling procedure [553] to train an underlying UNet [500]. The architecture also includes a text encoder, such as the Contrastive Language-Image Pre-training (CLIP) model [473] — either the original OpenAI CLIP [480] or its open-source counterpart, OpenCLIP [121, 290].

Stable Diffusion 2 (SD2)'s UNet has approximately 865 million trainable parameters; Stable Diffusion XL (SDXL) is larger, with 2.6 billion parameters and

has other advancements involving aspect ratio bucketing, micro-conditioning, and multiple text encoders and tokenizers. In terms of training data, SD models and OpenCLIP are both trained on subsets of the LAION-5B dataset [50, 523]. The exact training dataset for CLIP is unknown, but it is likely web-scraped data [480].

9.2.2 Copyright, reproducibility, and LAION datasets

LAION-5B is a dataset derived from a snapshot of the Common Crawl, a massive corpus of data scraped from the web. From this snapshot, the LAION organization curated pairs of image URLs and their corresponding alt-text captions for the intended use of training T2I and image-to-text (I2T) generative models [50, 523]. In practice, T2I models are typically trained on filtered subsets of the full LAION-5B dataset (e.g. LAION-2B [338]). Training T2I models on this dataset requires visiting the URLs and downloading the associated images. There are two elements of LAION datasets that are relevant to our work:

Copyright. The images associated with LAION datasets have unclear *provenance*: it is often not known what the original image sources are [351]. Although LAION datasets are released under the open MIT license, some experts note that it is unclear if this is sufficient to allow for training on the underlying images and captions, which often have their own copyrights [142, 272, 349–351]. Courts have not yet decided if training on these datasets is "fair use" — an important exception in copyright [349, 350, 363, 515, 551]. There are several copyright lawsuits for the alleged use of LAION-5B subsets to train generative models [24, 233, 301, 599, e.g.].

Reproducibility. Since LAION datasets only contain the image URLs, and not the images themselves, they are plagued with *link rot* [339].[1] When accessing LAION-5B, there is no guarantee the images still exist at their URLs, making it impossible to fully reproduce the dataset and opening up the possibility of data poisoning attacks [106]. A natural alternative is to not use LAION datasets for training. Instead, one could independently curate a dataset of CC-licensed images with known provenance that explicitly allow for copying, adaptation, and commercial use. As constituent images can be stored and distributed, this would also solve the link-rot problem, enabling greater reproducibility. (Further, LAION datasets are no longer public because they contain CSAM [63, 577].) We defer our discussion of sourcing CC-licensed images to Section 9.4, where we detail CommonCatalog: our new, open dataset. While CC images are an attractive alternative to LAION-5B, we note that CC images rarely contain the captions necessary to train T2I models. Therefore, we first need a method for captioning CC images.

9.3 Transfer Learning for Image Captioning

Our solution for handling the lack of captions in CC images is an intuitive type of transfer learning for producing high-quality synthetic labels. We describe this method, and note that there are various similar methods in prior literature on generative modeling. Altogether, these methods indicate that this type of transfer learning has become an increasingly common pattern: producing synthetic labels that later serve as inputs to training other generative models. We therefore give this method a shorthand name: *telephoning*.

[1] This also applies to other web-scrapes, e.g., DataComp [219].

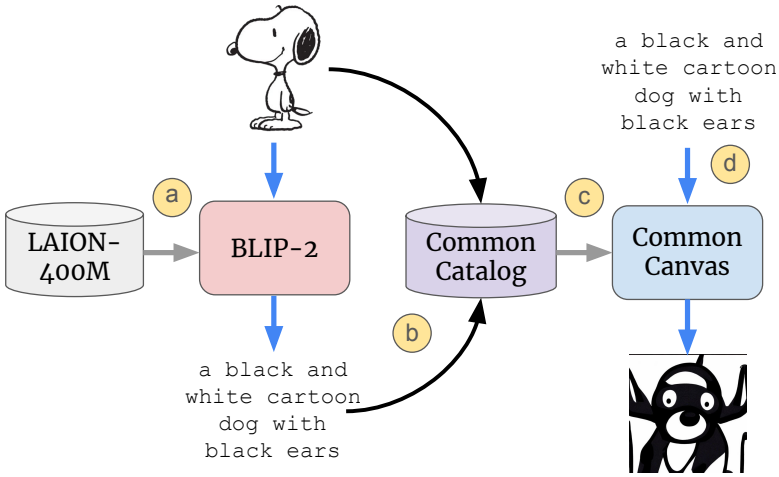


Figure 9.3: (**a**) We use the LAION-400M-pre-trained, I2T BLIP-2 model to produce synthetic captions for our uncaptioned CC images (e.g., the Wikipedia CC-licensed image of Snoopy). The synthetic captions are "lossy compressions" of the input images (e.g., `a black and white cartoon dog with black ears` has no mention of Snoopy). (**b**) We compile the resulting synthetic image-caption pairs into *CommonCatalog*, which (**c**) we use to train our open, T2I *CommonCanvas* models. (**d**) When we supply "lossy" captions to a T2I model, like a game of telephone, **it produces outputs that no longer resemble the original images** (e.g., CommonCanvas produces an image that matches the caption, but does not look like Snoopy).

9.3.1 Telephoning

Telephoning (Figure 9.3) proceeds in two steps. First, shown in Figure 9.3b, it takes inputs from a high-dimensional modality (e.g., images) and effectively performs a "lossy compression" to a (scarce) low-dimensional modality (e.g., short-text captions). Second, shown in Figure 9.3d, it takes the "lossy compression" and decompresses back to the high-dimensional modality. Because the intermediate compression step is "lossy," the ultimate output often does not remotely resemble the original input, just like a game of telephone [393]. We derive the term telephoning from the above intuition and use it as shorthand to denote instances of transfer learning that solve data-scarcity problems in multi-

modal generative modeling.

In this work, CC images are the high-dimensional inputs, and we use a pre-trained BLIP-2 model [369] for "lossy compression" to short-text captions (Figure 9.3a). Together, these CC-image-caption pairs comprise the CommonCatalog dataset (Section 9.4), which we use to train our CommonCanvas T2I models (Figure 9.3b). While BLIP-2 was pre-trained on LAION-400M [522], we emphasize that, for training CommonCanvas, we only ever have access to the captions — to the "lossy compressions" it produces. We never have direct access to LAION-400M or, importantly, anything that is similar to the images that BLIP-2 was trained on. Instead, we only have access to the mapping in the model, which, given an image input, produces "lossy" output text.

Telephoning & Copyright. We defer to experts about fair use (Section 9.2.2) — namely, regarding models like BLIP-2, and LAION-5B's images and alt-text captions. Generally, these experts seem to think that many cases will fall under fair use [349, 357, 515], especially when model outputs do not resemble their inputs (i.e., the use is "non-expressive" or "non-consumptive" [142]). This is the case with our use of BLIP-2 to produce "lossy" captions.

Nevertheless, it is possible that BLIP-2 could produce captions that resemble those in its LAION training data. This might seem to present a copyright concern similar to those that others have expressed about T2I generations that resemble LAION images. However, according to the U.S. Copyright Office, short phrases (like captions) may often not be copyrightable: "short phrases" often contain "an insufficient amount of authorship" to meet the threshold for copyright protection [575]. So, even if hypothetically BLIP-2 were to regurgitate captions from LAION verbatim, according to legal experts [349], the copyright

considerations are likely to be different than they are for generated images or generated long-form text. We defer to experts for more precise legal arguments, but note that this is another reason why we believe it is reasonable for us to rely on BLIP-2 for captioning our CC images.

9.3.2 Related work on telephoning

Our work aligns with the trend of using advanced generative models to address data scarcity. This is evident in various modalities, such as producing audio captions from image-text pairs [627] and text from audio [481]. Similar approaches have also been used to generate instruction-tuning datasets for both text and images [371, 377]. Concurrent work, e.g. LLaVA [377], has used visual question-answer models to augment existing caption datasets, such as the ones used in training DALLE·3 [56] and Chen et al. [120]. Our model is one of the first works to train on a dataset without any ground-truth captions, and one of the first to release our dataset along with a fully trained diffusion model. The caption upsampling approaches described in these other works could be used to further improve the captions of CommonCatalog in future work.

Captioning models have also been used to create descriptive captions to guide a diffusion model to create an image visually similar to a specific image In concurrent work, SynthCap [95] generates a synthetic captioning dataset using a diffusion model to generate images from captions — the inverse of our problem statement. We coin the term telephoning to short-hand processes like these, which include our work and prior work, and which we believe will become more prevalent as generative-model capabilities advance.

9.4 A CC-Image, Synthetic-Caption Dataset

We now introduce our open dataset, *CommonCatalog*. First, we describe the collection and curation process for the open-licensed, CC images. This process brings to light two challenges: caption-data incompleteness and image-data scarcity. To address the lack of CC captions, we show concretely how we use telephoning to produce high-quality synthetic captions to accompany our set of curated images. We investigate the topic of data scarcity in the next section, where we also discuss necessary systems-level training optimizations that enable efficient model iteration.

9.4.1 Sourcing licensed images for CommonCatalog

We focus on locating high-resolution Creative-Commons images that have open licenses. We began with the YFCC100M dataset, which consists of 100 million CC-licensed images and multimedia files, as well as Flickr IDs linking to the original data [579]. The images in the dataset associated with the original paper exhibit two issues that make it ill-suited for direct use to train Stable Diffusion: they are low-resolution, and many of them have licenses that do not expressly allow for the distribution of derivative works — a use that is in unsettled copyright law in the context of model training [349].

We therefore re-scraped these images from Flickr, based on the IDs provided in the YFCC100M metadata. Our scraped images are of very high resolution (exceeding 4K), which makes them more suitable for T2I training.

We exclude images with non-derivative (ND) licenses. The remaining im-

Figure 9.4: CommonCatalog-C contains images licensed only for commercial use; -NC contains -C as well as images licensed for non-commercial use.

Dataset	# Images	% Alt Text
CommonCatalog-C	26,232,417	30.76%
CommonCatalog-NC	67,015,331	31.22%

ages can be further divided into those that can be used for commercial (C) purposes and those that cannot (NC). As shown in Table 9.4, we accordingly construct two datasets, CommonCatalog-C and CommonCatalog-NC. We defer additional details about licenses to Appendix F.2, but emphasize that all of the included images have open licenses: individuals are free to use, adapt, and remix the images, so long as they attribute them. In total, CommonCatalog contains roughly 70 million images that can be used non-commercially, of which a approximately 25 million images can also be used commercially.

Directly sourcing CommonCatalog avoids some concerns (Section 9.2.2); however, it also comes with its own challenges. For one, CC images rarely have the alt-text captions necessary to train a T2I model like Stable Diffusion (Figure 9.4); those that do have associated text often just include the image title or a URL. For another, we could *only* find roughly 70 million usable CC images, which pales in comparison to the billions of images in LAION used to train SD2 (Section 9.5). We take each of these challenges in turn. First, in the next subsection, we show how we instantiate telephoning (Section 9.3) to produce high-quality, synthetic captions for CC images.

Source	Caption
Alt-Text (LAION-2B)	`"Latest 1PC Transparent Gradient Color Voile" "Window Curtain"`
BLIP2-OPT-2.7B	`"A living room with a white couch and curtains"`

Figure 9.5: Original vs. BLIP-2-generated captions for an image from LAION-2B. In this example. BLIP-2's caption better aligns with what a human would write. See Appendix F for more examples.

9.4.2 Synthesizing captions with telephoning

We compared several captioning models and chose the pre-trained BLIP-2 OPT2.5B model for synthesizing CommonCatalog's captions [369], based on qualitative analysis and state-of-the-art performance on MS COCO. BLIP-2 consists of three components: a pre-trained, frozen (i.e., fixed) visual encoder, a learned transformer network that converts the visual embeddings into a text prompt, and a frozen large language model (LLM) that takes in the prompt. The only trainable variables in the transformers are between the frozen visual encoder and the frozen LLM layers.

Given a LAION-2B image as input, we found that the resulting BLIP-2 caption is often qualitatively more descriptive than the corresponding LAION-2B ground-truth alt-text caption. LAION-2B captions often contain product names, irrelevant details, or poor grammar and syntax (Figure 9.5). This finding is corroborated by Nguyen et al. [443], which quantitatively shows that (in terms of CLIP Score) BLIP-2 captions are higher quality than ground-truth captions, at the cost of caption diversity. Based on these preliminary results, we captioned all of the YFCC100M Creative-Commons images, which required about 1,120 GPU A100 hours. We center-cropped and resized all of the images to a maxi-

mum size of 512x512 pixels, since captioning images at native resolution would be very expensive. At training time for CommonCanvas models, we use the high-resolution images.

We release our commercial (CommonCatalog-C) and non-commercial (CommonCatalog-NC) CC-image and synthetic-caption datasets with associated data cards. As an evaluation set, we also release the BLIP-2 captions that we produced for the non-derivative (ND) CC images that we did not use for training.

9.5 Optimizations and Data-Scarcity Analysis

High-resolution CC images are indeed much less abundant than web-scraped images; however, it is unclear if this scarcity presents a problem for training. Prior work has not studied in depth how much data is actually necessary to train high-quality SD2 models. We set out to quantify this amount by training multiple SD2 models on differently-sized subsets of LAION-2B. However, training a single SD2 model, even with hundreds of GPUs, can take several days. So, to make our data scarcity analysis more tractable, we first implemented several efficiency optimizations.

9.5.1 Software and hardware speed-ups

Stability AI reports an estimated 200,000 A100 hours to train SD2 [559]. Depending on hardware, a single SD2 training run could take anywhere from a few weeks to over a month. We sought out multiple avenues to reduce this

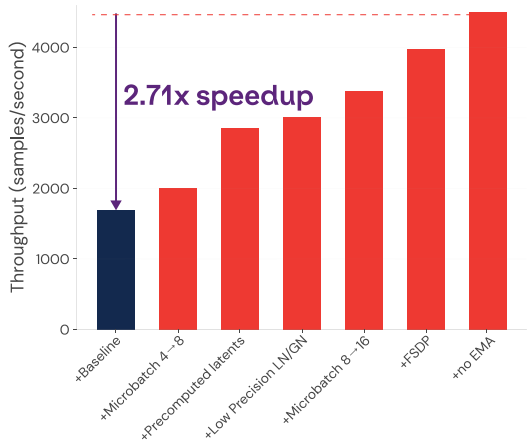


Figure 9.6: Cumulative effect of various speed-ups (totalling 2.71×) in our SD2 training pipeline evaluated on 128 A100s.

training-time constraint. We applied Flash Attention [163] with the xFormers library [353], pre-computed VAE and text encoder latents over the entire training dataset, cast all GroupNorm [626] and LayerNorm [31] to `float16` precision, and applied fully-sharded data parallelism (FSDP) to our training run. Finally we opted to only keep an exponential moving average of the weights for the final 3.5% of training. Altogether, we are able to achieve a 2.71X speedup in A100 hours over our SD2 baseline implementation.

We found that latent pre-computation helped the most at low resolutions, while FSDP also provided significant gains, especially at scale. The other optimizations helped reduce total memory usage, allowing us to increase the microbatch size for better hardware utilization. Figure 9.6 summarizes each of the proposed methods and the cumulative speedup that results from their application. Equipped with an optimized training setup, it is more feasible for us to study the effect of varying training-dataset size. More details can be found in Appendix F.4.

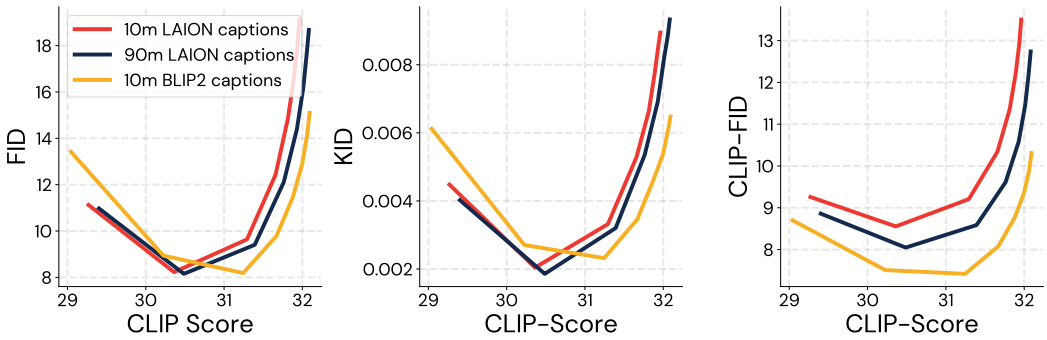


Figure 9.7: For different SD2 models trained on subsets of LAION (90M, 10M using either original captions or synthetic BLIP-2 captions), we compute FID [274], KID [62], CLIP-FID [337], and CLIP-Score [273] on 30K samples from MS COCO. We compute these metrics across a text-guidance scale of 1-8, with higher values indicating the model should respect the text prompt more. Lower FID, KID, and CLIP-FID indicate higher quality; higher CLIP-Score indicates higher quality. Together, these plots show that increasing the amount of training data from 10M to 90M samples does not lead to quantitative improvements. BLIP-2 re-captions provide nearly identical performance to LAION in terms of FID and KID; the re-captions indicate slightly better performance when using CLIP-FID as the quality metric.

9.5.2 Investigating data scarcity

YFCC100M contains 100 million images, about 10% the size of the 1.1B LAION examples we could access (due to link rot) — about 5% of the original LAION-2B dataset. An interesting question remains: *how much data is actually needed to train these diffusion models effectively; do we really need billions of images to get high-quality results?*

To answer this question, we train multiple SD2 architectures on increasingly smaller, random subsets of data from our LAION-1.1B dataset: 1.1B, 90M, 10M, and 1M sample subsets. While human evaluation remains the gold standard for evaluating generative models, we use proposed automated metrics like Frechet-Inception Distance [274], Kernal Inception Distance [62] and caption-alignment metrics such as CLIP Score [273] (Figure 9.6). We find that performance (FID and

KID on MS COCO) does not degrade until training with as few as 1 million images; our models trained on 10M and 90M subsets perform comparably to the entire 1.1B dataset (Appendix F, Figure F.1). Figure 9.7 further compares our SD2 variants trained on 10M and 90M LAION subsets across different guidance scales. We also plot the effect of using the original LAION captions vs. BLIP-2 synthetic captions at these size regimes (discussed further in Section 9.6.1). These findings suggest that SD2 models may be underparameterized. We hypothesize about why this might be the case and how much data is actually necessary to saturate the model in Appendix F.

9.6 Experiments

In this section, our model evaluations use automated, quantitative image-quality metrics from the literature. We measure performance with three automated image quality metrics on the commonly used MS COCO dataset [374]: Fréchet Inception Distance (FID) [274], Kernel Inception Distance (KID) [62], and CLIP-FID [337]. Each captures a slightly different measures of generated-image quality and diversity, in relation to statistics in the training data, with lower values corresponding to higher quality. Additionally, we evaluated CLIP-Score [273], which can help us understand the alignment between captions and their respective images, with higher values signaling better alignment. While these automated metrics are intended to be efficient proxies for human preferences in image quality, they often fall short; the gold standard for T2I model evaluation still remains human evaluation. Since synthetic captions differ so much from human-designed ones [443], we also set up a pairwise preference rating task to measure the relative quality of our trained models.

Figure 9.8: Using entirely Creative-Commons images and our synthetic captioning approach, we achieve comparable qualitative performance to public SD2, as seen in CommonCanvas generations, while only requiring a small fraction (< 3%) of the amount of training data. We include results for two CommonCanvas architectures, small (S) and large (L) (Section 9.6), and two CC-image datasets, commercial (C) and non-commercial (NC) (Section 9.4). We label our results accordingly as CommonCanvas-<architecture>-<dataset>.

9.6.1 Training with Synthetic Captions

First, we look at the effect of training with synthetic captions instead of ground-truth captions from LAION. Interestingly, we observe that synthetic captions can enhance the alignment of our model. For instance, the CLIP-Score for synthetic captions exceeded that of ground-truth captions as seen in Figure 9.7 (for CLIP-FID).

To get a more nuanced perspective on the effect of our synthetic captions, we assess CLIP-FID for image generations from different models on human- and computer-generated captions. In Figure 9.9, we compute CLIP-FID for various models trained using LAION, CommonCatalog, or LAION images re-captioned with BLIP-2; CLIP-FID is computed based on generating for prompts from MS COCO and the Conceptual Captions dataset. Unlike other caption datasets,

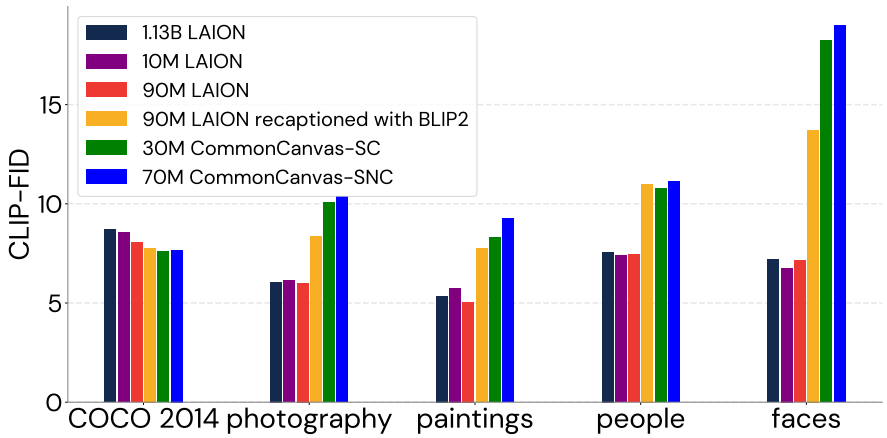


Figure 9.9: Evaluating models at 256 resolution on different subsets of the Conceptual Captions dataset and MS COCO. LAION models are trained on **1.1 billion**, **90 million (SD2-90M)**, and **10 million** subsets. We also train a model with **a 90 million subset re-captioned with BLIP-2** to evaluate distribution shift. The last two models are trained on **on the CommonCatalog-C**, and **CommonCatalog-NC**. We observe a domain shift between MS COCO and web-scraped Conceptual Captions. CLIP-FID may exhibit a preference for SD2 models, given that CLIP has been trained on a text style akin to that found in LAION. Subsampling the LAION dataset from 1.13B to 10M images does not seem to affect quantative performance. Using synthetic captions causes a significant performance drop on the LAION dataset when evaluated on Conceptual Caption test datasets, but not MS COCO.

MS COCO captions are human written. Most captions from web-based datasets (like LAION) are computer-generated [443]. BLIP-2 captions are also generated, but the BLIP-2 model is then fine-tuned to align with human-written captions. Given the higher quality of our synthetic captions, it is unsurprising that CommonCanvas's CLIP-FID is better (i.e., lower) for MS COCO (i.e., aligns better with human-written captions).

However, like any model, ours has limitations. CommonCanvas underperformed in several categories, including faces, general photography, and paintings. These datasets all originated from the Conceptual Captions dataset [536], which relies on web-scraped data. These web-sourced captions, while abundant, may not always align with human-generated language nu-

Figure 9.10: We compare CommonCanvas-S-NC (Ours) to SD2. Our model is less likely to generate iconic characters given suggestive prompts (drawn from Lee et al. [349]).

ances [56, 95, 443]. Although transitioning to synthetic captions introduces certain performance challenges, the drop in performance is not as dramatic as one might assume. Moreover, we speculate that the model will perform better if users provide their more specialized datasets to the model, such as FFHQ [311].

9.6.2 CommonCanvas vs. LAION-trained SD2

Given that our data-scarcity analysis suggests that CommonCatalog is large enough to train a high-quality SD2 model and that synthetic captions can perform well (Figure 9.6.1), we train two different CommonCanvas models: one trained on commercial (CommonCatalog-C) images, another on non-

commercial (CommonCatalog-NC). For a fair comparison with SD2, we use the OpenCLIP text encoder. Like BLIP-2, OpenCLIP is trained on LAION captions (Figure 9.2.2). For example generations, see Figure 9.8.

We also note that, although we train on Creative-Commons images, it is still possible for an adversarial prompt to produce content that includes iconic characters. In Figure 9.10, we subject our model to ambiguous prompts that are suggestive of such characters. Examples include visuals closely resembling Elsa from Frozen, Indiana Jones resembling Harrison Ford, and even a likeness to Harry Potter. Qualitatively, our model deviated more from these characters than SD2.

9.6.3 Reaching SD2 quality with CommonCanvas-L

We also did a human study measuring pairwise preference ratings for the 512x512 resolution CommonCanvas models compared to SD2 (Figure 9.12). In this experiment, human raters were shown a prompt (selected randomly from the PartiPrompts prompts set [636]) along with two generated images in randomized order, one from the reference model (public SD2) and the other from a CommonCanvas model. We report the fraction of the time users selected the image generated by the CommonCanvas model over the corresponding generation from SD2 as the user preference rate for that model. We find that our CommonCanvas models are slightly less preferred than SD2-90M, with preference rates of 37% for CommonCanvas-S-C and 38% for CommonCanvas-S-NC, which we find surprisingly high considering the smaller and synthetic nature of the dataset. Figure 9.8 displays the results from our human study.

Figure 9.11: Using CommonCanvas-SNC (Ours) to generate celebrities. Our model is worse at synthesizing individual people than SD2, but is capable of generating some noteworthy public figures. This result demonstrates how our model struggles to generate specific celebrities, which may be desirable from a privacy perspective.

Our previous results suggest that SD2 may be underparameterized. We additionally train a larger variant of CommonCanvas-N-C (CommonCanvas-L-NC) that has a significantly larger U-Net (the U-Net architecture from SDXL (Podell et al. [473], Appendix F). When we use CommonCanvas-L-NC, we achieve competitive performance with SD2 on user preferences (Figure 9.8). For the largest model, CommonCanvas-L-NC, we do not measure a statistically significant difference in user preference between this model and SD2.

9.7 Discussion and Related Work

In this paper, we train the CommonCanvas family of text-to-image, latent diffusion models using only Creative-Commons images and synthetic captions. We discuss and address data incompleteness and scarcity issues associated with

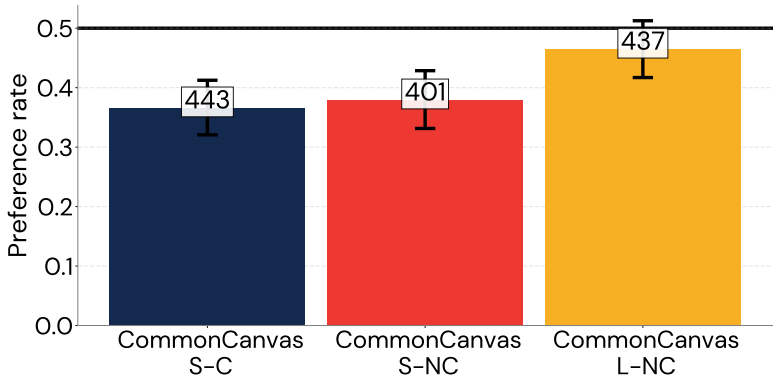


Figure 9.12: User preference study using Parti prompts. Preference rate (compared to SD2, the thick black horizontal line). CommonCanvas-L-NC matches the performance of SD2.

CC images. For data incompleteness, we propose telephoning, an intuitive type of transfer learning (Section 9.3), which we instantiate with BLIP-2 to produce synthetic captions for CC images (together, the CommonCatalog dataset; Section 9.4). Regarding data scarcity, we hypothesize that only a small fraction of the data contained in LAION-2B is actually necessary to saturate SD2, and that the examples in CommonCatalog should be sufficient for training. To make testing this hypothesis more efficient, we implement a variety of ML-systems optimizations, which achieve a 2.71× speed-up over our SD2 baseline.

Ultimately, we find that we can train the SD2 model on <3% of LAION-2B (i.e., roughly 70 million images; Section 9.5), yielding a model we call SD2-90M. This encourages us to train on CommonCatalog's commercially usable (also roughly 70 million) and non-commercially usable (roughly 25 million) examples. Compared to SD2, our CommonCanvas models under-perform in some categories, like faces, but CommonCanvas-L-NC demonstrates statistically equivalent performance with SD2 on human evaluation (Section 9.6).

While several recent works similarly address ML topics relating to copy-

right, the literature tends to concern text-to-text training data [414], be primarily theoretical [518, 605], involve ablation studies [335], or only handle verbatim memorization [109, 435] through the use of generation-time content filters [235], which has been shown to be an incomplete solution [294]. To the best of our knowledge, no prior open work attempts to train T2I models on only open-licensed data. Most prior work on image-caption-dataset creation has extracted caption data from Common Crawl [169, 219, 347]. We instead focus on synthesizing captions directly by using a pre-trained BLIP-2 model. Nguyen et al. [443] demonstrates that existing caption datasets can be improved by using BLIP-2 to replace low-quality image captions (e.g., in Datacomp), but does not focus on creating a new dataset of synthetic captions, as we do here.

Another limitation is that the YFCC100M data is about a decade old; its CC images are not as current as those in LAION-2B. In the future, we plan to augment CommonCatalog with Creative-Commons images from other sources, as well as test larger model architectures and more advanced captioning models, like LLaVA [377].

CHAPTER 10

TALKIN' 'BOUT AI GENERATION: COPYRIGHT AND THE GENERATIVE-AI SUPPLY CHAIN (THE SHORT VERSION)

Memorization (Chapter 8) and licensed training data (Chapter 9 are just two of many issues that generative AI presents for copyright. In this chapter, we explore these issues in more detail. However, this chapter is a much-abridged version of more extensive work published on this topic [349]. This work has had a significant impact on legal scholarship, U.S. and U.K. AI policy, and more.

Chapter summary: "Does generative AI infringe copyright?" is an urgent question. It is also a difficult question, for two reasons. First, "generative AI" is not just one product from one company. It is a catch-all name for a massive ecosystem of loosely related technologies. These systems behave differently and raise different legal issues. Second, copyright law is notoriously complicated, and generative-AI systems manage to touch on a great many corners of it. They raise issues of authorship, similarity, direct and indirect liability, and fair use, among much else. These issues cannot be analyzed in isolation, because there are connections everywhere.

We aim to bring order to the chaos. To do so, we introduce the **generative-AI supply chain**: an interconnected set of stages that transform training data (millions of pictures of cats) into generations. (a new, potentially never-seen-before picture of a cat that has never existed). Breaking down generative AI into these constituent stages reveals all of the places at which companies and users make choices that have copyright consequences. It enables us to trace the effects of upstream technical designs on downstream uses, and to assess who in these complicated sociotechnical systems bears responsibility for infringement when

it happens. Because we engage so closely with the technology of generative AI, we are able to shed more light on the copyright questions. We identify the key decisions that courts will need to make as they grapple with these issues, and point out the consequences that would likely flow from different liability regimes.

This chapter is a licensed derivative copy of work published and awarded a Long Presentation slot at *ACM CSLAW 2024* [350], which is a much-abbreviated version of a law review article published at *The Journal of the Copyright Society* [349].

10.1 Introduction

Generative-AI systems like ChatGPT, Gemini, DALL·E, and Ideogram can turn a user-supplied prompt like "`give three arguments why marbury v. madison was wrongly decided`" into a persuasive essay, or "`a cowboy riding a rocket ship`" into a work of digital art. They are unpredictable and complex; they break out of existing legal categories. In particular, because generative-AI systems are trained on millions of examples of human creativity, they raise serious copyright issues. These copyright issues have not gone unnoticed. Numerous plaintiffs have sued leading generative-AI companies for copyright infringement, with potential damages reaching into the billions of dollars.

This chapter looks systematically at how copyright applies to generative-AI systems. Our first contribution is to be precise about what "generative AI" is. It is not just one product from one company. Instead, it is a catch-all term for a

massive ecosystem of loosely related technologies, including conversational text chatbots like ChatGPT, image generators like Midjourney and DALL·E, coding assistants like GitHub Copilot, and systems that compose music, create videos, and suggest molecules for new medical drugs. Generative-AI models have different technical architectures and are trained on different kinds and sources of data using different algorithms. Some take months and cost millions of dollars to train; others can be spun up in a weekend. Some models are offered through paid online services; others are distributed open-source, such that anyone could download and modify them.

We take the complexity and diversity of generative-AI systems seriously. We introduce the **generative-AI supply chain**: an interconnected set of stages that transform training data (millions of pictures of cats) into generations (a picture that may never have been seen before of a cat that may not exist). We conceive of eight stages: 1) production of creative works, 2) conversion of creative works into quantified data, 3) creation and curation of training datasets, 4) base model (pre-)training, 5) model fine-tuning to adapt to a specific problem domain, 6) model release or deployment within a software system, 7) generation, i.e., the AI-assisted production of new creative works, and 8) alignment, i.e., adjusting the model and system to advance goals (such as helpfulness, safety, legal compliance). The supply chain is not a simple cascade from data to generations. Instead, each stage is regularly adjusted to better meet the needs of the others. Breaking down generative AI into these constituent stages reveals all of the places at which companies and users make choices that have copyright consequences.

We then work systematically through the copyright analysis of these differ-

ent stages. Copyright law is notoriously complicated, and generative-AI systems manage to touch on a great many corners of it. They raise issues of authorship, similarity, direct and indirect liability, fair use, and licensing, among much else. These issues cannot be analyzed in isolation, because there are connections everywhere. Whether the output of a generative-AI system is fair use can depend on how its training datasets were assembled. Whether the creator of a generative-AI system is secondarily liable can depend on the prompts that its users supply. We trace the effects of upstream technical designs on downstream uses, and assess who in these complicated sociotechnical systems bears responsibility for infringement when it happens. Because we engage so closely with the technology of generative AI, we are able to shed more light on the copyright questions. We do not give definitive answers as to who should and should not be held liable. Instead, we identify the key decisions that courts will need to make as they grapple with these issues, and point out the consequences that would likely flow from different liability regimes.

We proceed in three parts. We:

- Describe the generative-AI supply chain in detail, including what happens at each stage, the diversity of variations on the basic theme, and the design choices that the various actors must make to create and use a generative-AI system (Section 10.2).
- Provide examples of how the supply-chain framing facilitates detailed copyright analysis, covering substantial similarity, direct infringement, and fair use. We ask *what* might possibly be an infringing technical artifact, *who* might be an infringing actor, and *when* infringement may occur, and discuss how the choices made by actors at one point in the supply

chain affect the copyright risks faced by others (Section 10.3).

- Detail broader lessons, including the options courts have and how they should conceptualize generative AI (Section 10.4).

Altogether, we argue that copyright pervades the generative-AI supply chain, that fair use is not a silver bullet, that the ordinary business of copyright litigation will continue even in a generative-AI age, and that courts should beware of metaphors that provide too-easy answers to the genuinely hard problems before them. This chapter is a shortened version of a law review article, which treats these topics in much greater detail [349].

10.2 The Generative-AI Supply Chain

We assume introductory familiarity with machine learning (ML) and generative AI, and delve right into our discussion of the generative-AI supply chain. To begin, we note that one of the big enablers of today's generative-AI systems is scale. Notably, scale complicates *what* technical and creative artifacts are produced, *when* these artifacts are produced and stored, and *who* exactly is involved in the production process. In turn, these considerations are important for how we reason about copyright implications: *what* is potentially an infringing artifact, *when* in the production process it is possible for infringement to occur, and *who* is potentially an infringing actor [146].[1]

[1]The generative-AI supply chain is a very good example of the "many hands" problem in computer systems. That is, there are many diffuse actors, at potentially many different organizations, that can each have a hand in the construction of generative-AI systems. It can be very challenging to identify responsible actors when these systems transgress broader societal expectations — in our case, the preservation of copyrights. See Cooper et al. [146, pp. 867-869] (describing the problem of "many hands" in data-driven ML/AI systems);

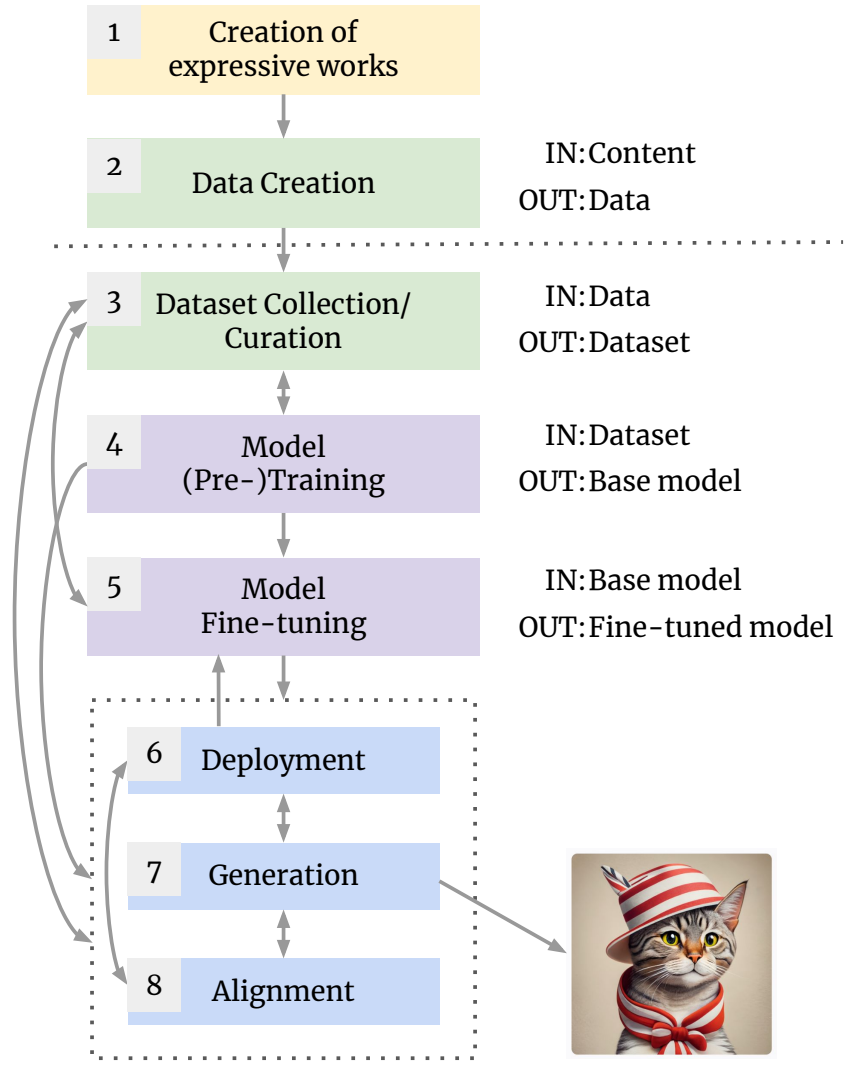


Figure 10.1: The generative-AI supply chain. We map out eight stages: 1) creation of expressive works, 2) data creation, 3) dataset collection/curation, 4) model (pre-)training, 5) model fine-tuning, 6) system deployment, 7) generation, and 8) model alignment. The creation of expressive works and data creation pre-date the advent of today's generative-AI systems (dotted line). There are many possible ways to connect the other six stages. Deployment, model alignment, and generation tend to happen in concert (dotted box). Generations can be used as training data (arrow from generation (7) to dataset collection/curation (3)). In this case, generation serves simultaneously as the creation of expressive works (1) and data creation (2). Curated data examples can be used for retrieval-augmented generation (arrow from dataset collection/curation (3) to generation (7)). APIs in deployed service can be used to do custom fine-tuning (arrow from deployment (6) to fine-tuning (5)).

To provide some structure for reasoning about this complexity, which will facilitate our copyright analysis in Section 10.3, we introduce our abstraction for reasoning about generative AI as a supply chain. We conceive of the **generative-AI supply chain** as having eight stages (see Figure 10.1): the creation of expressive works (Section 10.2.1), data creation (Section 10.2.2), dataset collection and curation (Section 10.2.3), model (pre-)training (Section 10.2.4), model fine-tuning (Section 10.2.5), system deployment (Section 10.2.6), generation (Section 10.2.7), and model alignment (Section 10.2.8). Each stage gathers inputs from prior stage(s) and hands off outputs to subsequent stage(s), which we indicate with (sometimes bidirectional) arrows.

The first two stages, the creation of expressive works and data creation, predate the advent of generative-AI systems. Nevertheless, they are indispensable parts of the production of generative-AI content, which is why we begin our discussion of the supply chain with these processes. The following six stages reflect processes that are new for generative-AI systems. The connections between these supply-chain stages are complicated. In some cases, one stage clearly precedes another (e.g., model pre-training necessarily precedes model fine-tuning), but, for other cases, there are many different possible ways stages can interact, and they may involve different actors. We highlight some of this complexity in the following subsections.

10.2.1 The Creation of Expressive Works

Artists, writers, coders, and other creators produce expressive works. Generative-AI systems do, too;[2] but state-of-the-art systems are only able to do so

[2]We discuss this in more detail below with respect to generation (Section 10.2.7).

because their models have been trained on data derived from pre-existing creative works.[3] It is worth remembering that, historically, the production of most creative works has had nothing to do with ML.[4] Painters have composed canvases, writers have penned articles, etc. without considering how their works might be taken up by automated processes. Nevertheless, these works can be transformed into quantified data objects that can serve as inputs for ML. They can be easily posted on the Internet and circulated widely, making them accessible for the development of generative-AI systems. As a result, authors and their works are a part of the generative-AI supply chain, whether they would like to be or not (Figure 10.1, stage 1).

10.2.2 Data Creation

Original expressive works are distinct from their datafied counterparts.[5] Data examples are constructed to be computer-readable, such as the JPEG encoding of a photograph. For the most part, the transformation of creative content to data formats predates generative AI (Figure 10.1, stage 2), but all state-of-the-art generative-AI systems depend on it. They rely on data that coheres with their underlying models' respective modalities: text-to-text generation models are trained on digitized text, text-to-image models are trained on both text and im-

[3] A data example is not the same as the expressive work. Additionally, some models are trained on synthetic data, typically generated by other generative-AI models [236, e.g.]. However, training predominantly on synthetic data is not reflective of current common practices in today's generative-AI systems. Further, there are concerns that training on synthetic data can seriously compromise model quality. See generally Shumailov et al. [539] (detailing "model collapse" in different generative models).

[4] It appears increasingly likely that some content will be created specifically for model training. For example, hiring photographers to take photographs specifically for model training. Companies like Scale AI already create content (in the form of labels and feedback) specifically for the purpose of training models [516].

[5] Of course, data examples can still be copies of original works, and thus still infringe intellectual property rights.

ages, text-to-music models are trained on text and audio files, and so on. This is an important point for our purposes because works that have been transformed into data have been fixed in a tangible medium of expression, and hence are subject to copyright.[6] In turn, generative-AI systems are often are trained on data that include copyrighted expression. The GitHub Copilot system involves models trained on copyrighted code,[7] ChatGPT's underlying models are trained on text scraped from the web, Stability AI's Stable Diffusion is trained on text and images, and so on. For the most part, it is the copyright owners of these datafied individual works who are the potential plaintiffs in a copyright infringement suit against actors at other stages of the supply chain (Section 10.3).

10.2.3 Dataset Collection and Curation

The training process for cutting-edge generative-AI models requires vast quantities of data. Dataset creators often meet this need by scraping the Internet.[8] This process involves numerous curatorial choices, including filtering out material that creators do not want to include, such as "toxic speech" [351].[9] Dataset creators are also necessarily curators.[10]

[6]We discuss fixation in Section 10.3.1. An exception is training data produced by generative-AI systems, as such data currently have been found to not be copyrightable. See *Thaler v. Perlmutter* [574]. We discuss using generations as training data in Section 10.2.7.

[7]Until recently, Copilot was built on top of OpenAI's Codex model.

[8]This is not the only way to collect large amounts of data. See Lee et al. [351] (discussing other ways datasets may come to be).

[9]See generally Lee et al. [351] (discussing dataset creation and curation choices, including toxic content filtering).

[10]This is why we choose to place creation and curation as the same stage in the pipeline. Note, however, that creation and curation do not *always* have to happen together, and may involve different sets of actors. It is also possible for curation to happen after the start of model training, in response to metrics that are observed during the training process. That is, curation could follow (and then also precede further) model (pre-)training (Figure 10.1, stage 4), or model fine-tuning (Figure 10.1, stage 5).

With respect to the generative-AI supply chain, there are several points worth highlighting in dataset collection and curation processes (Figure 10.1, stage 3). First, while dataset creation and curation can be carried out by the same entities that train generative-AI models, it is common for them to be split across different actors. The Stable Diffusion model, for example, is trained on images from datasets curated by the non-profit organization LAION.[11] It is necessary, therefore, to consider the liability of dataset creators separately from the liability of model trainers.

Second, dataset curation will frequently involve "the collection and assembling of preexisting materials or of data that are selected, coordinated, or arranged in such a way that the resulting work as a whole constitutes an original work of authorship" [152]. Thus, training datasets can themselves be copyrighted; copying of the dataset *as a whole* without permission could constitute infringement, separate and apart from infringement on the underlying works.[12]

Third, while a few training datasets include metadata on the provenance of their constitutive data examples, many do not. Provenance makes it easier to answer questions about the sources a model was trained on, which can be relevant to an infringement analysis. It also bears on the ease with which specific material can be located, and if necessary removed, from a dataset. However, the use of web-scraping to collect generative-AI training datasets makes provenance difficult to track [351]. Even if a dataset creator releases the dataset itself under a license, this does not guarantee that the works in the dataset are appropriately li-

[11]Technically, LAION presents the dataset as a collection the URLs of the images. Model trainers visit each URL to collect images for training.

[12]In practice, however, it appears that most uses of training datasets are licensed — either through a bilateral negotiation or by means of an open-source license offered to the world by the dataset compiler.

censed,[13] as is currently up for debate with the LAION-5B dataset [24, 50, 523].[14]

10.2.4 Model (Pre-)Training

Following the collection and curation of training datasets, it is possible to train a generative-AI model. A model trainer[15] (Figure 10.1, stage 4) selects a training dataset, a model architecture (i.e., a set of initialized model parameters), a training algorithm, and a seed value for the random choices made during the training.[16] The process of transforming these inputs into a trained model is expensive. It requires a substantial investment of multiple resources: time, data storage, and compute. For example, BLOOM (a 176-billion-parameter open-

[13]Indeed, the creators would have to check that they have abided by each data example's respective license. Some example pairs could potentially have multiple licenses – e.g., an image and its associated caption could have their own copyrights and licenses.

[14]LAION-5B, a large image-caption dataset, was released as under Creative Commons CC-BY 4.0. LAION-5B released a dataset of text captions and URLs to images, instead of the images themselves [50, 523]. It is unclear if the LAION team had the rights to license the images within. Notably, the website introducing the LAION dataset provides a feature called "pwatermark," which is a prediction of how likely the image is to contain a watermark. The LAION team estimates that the 6.1% of the dataset Laion2B-en contains watermarked images. Another example comes from the complaint in *Tremblay v. OpenAI, Inc.* [583], which alleges that ChatGPT's underlying model(s) were trained on datasets that do not license the books data that they contain. The complaint alleges that the training data included books from infringing "shadow libraries" like Library Genesis. See Complaint at *Tremblay v. OpenAI, Inc.* [583, p. 34] But this claim is based on circumstantial evidence, because the datasets it was trained on have not been made public. Text from books have been a key player in other dataset-related complaints. For example, The Pile data was originally released under the MIT license [59]. The Pile was core to the complaint in *Kadrey v. Meta Platforms* [304], since the Pile claimed to contain 108GB of the dataset Books3 (which itself contains content from Bibliotek, a popular torrent interface). The original download URL for The Pile (`https://the-eye.eu/public/AI/pile/`) is no longer resolving (as of September 2023). LAION has also been taken down from popular hosting services, following a report documenting the presence of CSAM at associated image URLs.

[15]We distinguish between the person or organization that trains from those that create the model architecture, as they may not be the same.

[16]ML uses tools from probability and statistics, which reason about randomness. However, computers are not able to produce truly random numbers. Instead, algorithms exist for producing a sequence of *pseudo*-random numbers. A random seed is an input to a pseudo-random number generator, which enables the reproduction of such a sequence. The trainer also selects hyperparameters [145], which we elide for simplicity.

source model from HuggingFace) was trained for 3.5 months, on 1.6 terabytes of text, using 384 GPUs [51, 624]; it cost an estimated $2-5 million.[17] As another point of reference, MosaicML has trained a GPT-3-quality model for less than $0.5 million.[18] Altogether, the dollar cost can range from six to eight figures.[19]

The output of the training process is typically called a **pre-trained model** or **base model**.[20] A base model has many possible futures. It could sit idly in memory, collecting figurative dust.[21] The model could be uploaded to a public

[17]Training costs are often not reported. Even when training cost is reported, development costs (including labor) are often omitted, despite being a critical (and often most expensive) part of overall model development.

[18]The original cost to train GPT-3 is unpublished, though, based on its size, is likely higher than $0.5 million. MosaicML reports to have trained a GPT-3-*quality* model. This means the model performs to a similar standard as GPT-3 does. Nevertheless, MosaicML's model is substantively different from GPT-3. For one, MosaicML's model is much smaller — 30 billion parameters compared with the original GPT-3 model's 175 billion. Additionally, MosaicML trained their model on more data, shifting some of the development cost toward data collection and away from model training. It is worth noting that GPT-3 was originally released two years before MosaicML's model was trained, and thus the MosaicML training process likely incorporated additional technological improvements. See generally Venigalla and Li [597] (regarding MosaicML's model). See generally Brown et al. [92] (for the size of GPT-3).

[19]Further, the training process is not completely automated; training often requires people to monitor and tweak the model. For example, model trainers typically run evaluation metrics on the model while it is being trained, in order to assess the progress of training. Google's TensorBoard [572] and software from Weights & Biases [617] are two tools for running evaluation metrics and monitoring during training. Depending on these metrics (which attempt to elicit how "useful" or "good" the model is, but are not comprehensive [351]) model trainers may pause the training process to manually revise the training algorithm (e.g., change the hyperparameters) or the dataset, which we indicate with bidirectional arrows at Figure 10.1, stages 3-4. Human intervention in response to metrics necessarily makes model training an iterative process.

[20]Others use the term "foundation model." The term "foundation" can be easily misunderstood. It should not be interpreted to connote that "foundation models" contain technical developments that make them fundamentally different from models produced in the nearly-a-decade of related prior work. The term itself has been met with controversy within the ML community, which can be seen expressed on programming forums and in conversations, e.g., we refer to a Twitter thread (and its associated offshoots) that involves renowned researchers and some of the Stanford authors that coined the term "foundation models." (*See* `https://twitter.com/tdietterich/status/1558256704696905728`).

[21]This reveals the murky line between what exactly is a program and what exactly is data in ML, more generally. The set of parameters can be viewed as a *data structure* containing vectors of numbers that, on its own, does not *do* anything. However, we could load that data structure into memory and apply some relatively lightweight linear algebra operations to produce a generation. In this respect, we could also consider the model to be a program (and, indeed, an algorithm). The model, if given a prompt input, can also be executed like a program. Note

server,[22] allowing others to download it and use it however they want.[23] The model could be integrated into a system and deployed as a public-facing application (Section 10.2.6), which others could use directly to produce generations (Section 10.2.7). Or, the model could be further modified by the initial model trainer, by another actor at the same organization, or, if made publicly available, a different actor from a different organization. That is, another actor could take the model parameters and use them as the input to do additional training with new or modified data. This possibility of future further training of a base model is why this stage of the supply chain is most often referred to as ***pre-training***, and why a base model is similarly often called a **pre-trained model**. Such additional training of the base model is called **fine-tuning**.

10.2.5 Model Fine-Tuning

Base models trained on large-scale, web-scraped datasets are not typically optimized to apply specialized domains of knowledge. For example, an English text-to-text base model may be able to capture general English-language semantics, but not able to reliably apply detailed scientific information about molecular biology.

This is where fine-tuning comes in (Figure 10.1, stage 5). Fine-tuning is the process of modifying a preexisting model and making it better along some di-

that the term "model" is overloaded; it can be used to refer to the model parameters (vectors of numbers) or to the model as a combination of software and the model parameters, which together can be executed like a program.

[22]For example, HuggingFace hosts a repository of over 300,000 open-sourced models [286].

[23]They could fine-tune the model (Section 10.2.5), embed the model in a system that they deploy for others to use (Section 10.2.6), produce generations (Section 10.2.7), align the model (Section 10.2.8), or do some subset of these other stages of the supply chain. From this example, we can see how the supply chain is in fact iterative, which we illustrate in Figure 10.1.

mension of interest. This process often involves training on additional data that is more aligned with the specific goals.[24] If we think of training as transforming data into a model, fine-tuning transforms a model into another model. Fine-tuning essentially involves just running more training. However, fine-tuning and pre-training may use different inputs, which ultimately makes the trajectories and outputs of their respective training processes very different.[25] To add more precision to our previous statement: fine-tuning transforms a model into another model, while incorporating more data.

Forks in the supply chain. Two important observations follow from our description of fine-tuning as (effectively) just performing more training. For one, a model trainer does not have to fine-tune at all. Prior to fine-tuning, there is a fork in the generative-AI supply chain with respect to the possible futures of the base model after pre-training (stage 4): One could take the output base model from pre-training, and use this model directly as the input for system deployment (stage 6), generation (stage 7), or model alignment (stage 8). Alternatively, it is possible to perform multiple separate passes of fine-tuning — to take an already-fine-tuned model, and use it as the input for another run of fine-tuning on another dataset.[26]

For each possibility, there can be different actors involved. Sometimes, the creator of a model also fine-tunes it. Google's Codey models (for code generation) are fine-tuned versions of Google's PaLM 2 model [242]. In other cases,

[24] And thus the reason for the bidirectional arrow between stages 3 and 5 in Figure 10.1. Similar to pre-training, monitoring metrics during fine-tuning may lead to further dataset curation (Section 10.2.4).

[25] There are other relevant factors in training, including choice of hyperparameters and choice of hardware. These, too, can change between pre-training and fine-tuning. We again elide these details for simplicity.

[26] In this respect, it is important to note that a model is a "base" or "fine-tuned" model *only in relation to other models*. These terms do not capture inherent technical features of a model; instead, they describe different processes by which a model can be created.

when a model's weights are publicly released (as Meta has done with its Llama family of models) [441, 581, 582], others can take the model and independently fine-tune them for particular applications. A Llama fine-tuner could release their model publicly, which in turn could be fine-tuned by another party.[27] To use a copyright analogy, a fine-tuned model is a derivative of the model from which it was fine-tuned; a repeatedly fine-tuned model is a derivative of the (chain of) fine-tuned model(s) from which it was fine-tuned.

It is helpful to make the base-/fine-tuned model distinction because different parties may have different knowledge of, control over, and intentions toward choices like which data is used for training and how the resulting trained model will, in turn, be put to use. A base-model creator, for example, may attempt to train the model to avoid generating copyright-infringing material. However, if that model is publicly released, someone else may attempt to fine-tune the model to remove these anti-infringement guardrails. A full copyright analysis may require treating them differently and analyzing their conduct in relation to each other (Section 10.3.4).

10.2.6 Model Release and System Deployment

It is possible to release a model or deploy it as part of a larger software system, use the model to produce generations (Section 10.2.7), or to take the

[27] To give a concrete example of the many actors in the generative-AI supply chain, consider Vicuna. LMSYS Org fine-tuned Meta's Llama model on the crowd-sourced ShareGPT dataset to produce Vicuna [534, 571]. ShareGPT is a crowd-sourced dataset composed of conversational logs of user interactions with ChatGPT. It contains both content created by users and by the generative-AI model embedded in ChatGPT (either GPT-3.5 or GPT-4, depending on the user) [534]. Vicuna has also released their model publicly, affording a potentially infinite host of actors the ability to fine-tune the model on additional data. See Raffel [485, slide 15] (for a figure showing many fine-tuned models building on one base model).

trained model and further alter or refine it via model alignment techniques (Section 10.2.8). In brief, there is a complicated interrelationship between the deployment, generation, and alignment stages. They can happen in different orders, in different combinations, and at different times for different generative-AI systems. For purely expository purposes, we present them one at a time, starting with **model release** and **system deployment** (Figure 10.1, stage 6).

A model is open-source **released** when its model parameters are uploaded to a server or platform (like HuggingFace [286]), from which others can download it.[28] Released models, which include Meta's Llama family of models [441, 581, 582] and Stable Diffusion [499] give others direct access to their parameters. Developers can write their own code to produce generations, or alter the model through fine-tuning or model alignment (Section 10.2.8).

In contrast, closed-source models are not directly available to external users. They are typically embedded in large, complex software systems, which are **deployed** to both internal and external users through software services. For example, a model could be hosted by a company (e.g., OpenAI, Stability AI, or Google). It could be used internally to support various services (e.g., Google has integrated an internally-developed LLM into Google Search), or released as a hosted service that gives external users access to generative-AI functionality.

External-facing services can be deployed in a variety of forms, and *do not* typically include the ability to change the model's parameters. They can be

[28] Meta first asked interested parties to request Llama's model parameters, rather than uploading them publicly on the web. However, Llama's model parameters were quickly leaked on the website 4chan [600]. This incident shows how challenging it can be to control access to models once released. Llama also includes a use policy in the Llama 2 Community License that outlines prohibited uses of the model. Of course, it is impossible to enforce prohibited uses when releasing model parameters. This is also why many model trainers choose to release models through hosted services. See Llama 2 [13] (for the Llama 2 Community License).

browser-based user applications (e.g., ChatGPT, Midjourney, DreamStudio), or public (but not necessarily free) APIs for developers (e.g., GPT models, Cohere).[29] Some model trainers provide a combination of release and deployment options. For example, DreamStudio is a web-based user interface [181] built on top of services hosted by Stability AI [14]; the DreamStudio application gives external users access to a generative-AI system that contains the open-source Stable Diffusion model [499], which Stability AI also makes available for direct download.[30]

This is a familiar spectrum from Internet law, from cloud-hosted services at one end to fully open-source software at the other, with closed-source apps in between. These deployment methods offer varying degrees of customization and control on the part of the deployer and the user. For example, a generative-AI system deployed as a service will often modify the user-supplied prompt before inputting it to the model. Several applications (e.g., ChatGPT, Gemini, and Sydney), add additional instructions ("application prompts") to the user's input to create a compound prompt [452, 645].[31] The additional instructions change the behavior of the model's output on a user prompt.[32] For example, compare the following two application prompts: "`I want you to act as an English translator,`

[29] Another deployment option is a command-line interface (CLI), which takes a user-supplied prompt as input (via a code terminal) and directly returns the resulting generation as output. `https://ollama.ai/` (the download link of the Ollama CLI, which is a wrapper program around various Llama-family LLMs).

[30] It is possible that models released and deployed in multiple ways might not all be exactly the same; they could have different versions of model parameters. This may be made explicit to users, as with ChatGPT, or may not be communicated to them, and thus unclear or unknown. See generally OpenAI [450] (regarding both GPT-3.5 and GPT-4 model integration into the ChatGPT web application).

[31] See generally Zhang and Ippolito [645] (which discovers proprietary system prompts); OpenAI [452] (announcing a ChatGPT feature that allows users to provide their own additional prompts, which get appended to their future inputs to create compound prompts).

[32] This kind of prompt transformation is another technique for steering the behavior of a model.

`spelling corrector and improver..."` and `"I want you to act as a poet. You will create poems that evoke emotions and have the power to stir people's soul..."` [17].[33]

Typically, model trainers and owners maintain the most control over models deployed through hosted services and the least over models released as model parameters [600]. By embedding a model within a larger system, they can imbue it with additional behaviors [144]. For example, APIs and web applications allow deployers to filters a model's inputs or outputs. For example, ChatGPT will often respond with some version of: "I'm really sorry, but I cannot assist you with that request," when its "safety" filters are tripped.[34] GitHub Copilot expressly states that it uses "filters to block offensive words in the prompts and avoid producing suggestions in sensitive contexts" [234]. Additionally, some services include output filters to avoid generating anything that looks too similar to a training example [235].[35] Unfortunately, output filtering is an imperfect process.(See Section 10.3.3).[36]

[33]See OpenAI [17] (These prompts and more can be found on this site); DAIR.AI [162] (This handbook provides an introduction to creating prompts for large language models); OpenAI [452].

[34]These filters may detect undesired inputs and prevent the model from generating an output, or detect undesired outputs and prevent the system from displaying the generation. In both cases, the model parameters would not be changed. This need not be the case, the model parameters may also be directly modified through alignment to respond to undesired inputs in a more desirable way. Of course, though, for ChatGPT, we do not know exactly how filters are implemented.

[35]See `https://news.ycombinator.com/item?id=33226515` (for related discussion on the Hacker News forum).

[36]Each mechanism for making model functionality widely available has different pricing structures that can ultimately impact the quality of the model. While the open-source community works hard to create and release models that compete with the best closed-source models, current open-source models are mostly trained on open-sourced data and are often lower quality. The best open-sourced models are very good, but still not as good as closed-source proprietary models. For example, Technology Innovation Institute in Abu Dhabi recently released the model, Falcon 180B (a 180 billion parameter model), which they claim is better than Meta's Llama 2 but still behind GPT-4 [293]. Additionally, differences between open- and closed-source datasets can lead resulting trained models to vary in quality. For example, Min et al. [414] uses public domain and permissively licensed text to train a language model, and demonstrates a

10.2.7 Generation

Generative-AI models produce output generations in response to input prompts.[37] While a few users produce generations from open-source models by writing code to interact with the model parameters to execute the generation process,[38] most users a interact with models only indirectly, through an API, web service, or application.

Users can affect generations in a few ways. First, there is the *prompt itself*. Some prompts, like `"a big dog"`, are simple and generic. Others, such as `"a big dog facing left wearing a spacesuit in a bleak lunar landscape with the earth rising as an oil painting in the style of Paul Cezanne"`, are more detailed. Second, there is the *choice* of which deployed system to use (which embeds an implicit choice of model). For example, a user that wants to perform text-to-image generation on a browser-based interface needs to select between Ideogram, DALL·E-2, Midjourney, and other publicly available text-to-image applications that could perform this task. A user typically selects an application with the outputs partially in mind, so that one choice or another can indicate an attitude towards the possibility of infringement. Users may also revise their prompt to attempt to create generations that more closely align with their goals. And, third, there is *randomness* in each generation.[39] It is typical, for example, for

degradation in quality in domains that are not well represented in the data. Additionally, data in the public domain can be unrepresentative of certain demographic groups [364].

[37] See Section 10.2.4 (noting, however, that models do not *have to* be used to produce generations).

[38] See Section 10.2.4 (discussing how the term "model" is overloaded, and can refer to model parameters being embedded in a program that executes (typically linear algebra) operations to to perform generation.)

[39] For generative models, there are many reasonable outputs for the input. There are also other sources of randomness in generation that are implementation-specific, such as the choice of decoding strategy for language models. See Riedl [495] (for an accessible discussion of decoding).

image applications to produce several candidate generations. DALL·E-2, Midjourney, and Ideogram all do this.

As we will see, characterizing the relationship between the user and the chosen deployed system is one of the critical choice points in a copyright-infringement analysis. There are at least three ways the relationship could be described:[40]

- The user actively drives the generation through choice of prompt, and the system passively responds. In this view, the user is potentially a direct infringer, but the application is like a web host, ISP, or other neutral technological provider.
- The system is active and the user passive. In this view, the user is like a viewer of an infringing broadcast, or the unwitting buyer of a pirated copy of a book. Primary copyright responsibility lies with the deployed system, and possibly with others further upstream in the generative-AI supply chain.
- The user and system are active partners in generating infringing outputs. In this view, the user is like a patron who commissions a copy of a painting; the system is like the artist who executes it. They have a shared goal of creating an infringing work.

We will argue that there is no universally correct characterization. Which of these three is the best fit for a particular act of generation will depend on the system, the prompt, how the system is marketed, and how users can interact

[40]We focus on deployed systems — and their API and web-based interfaces — because there are more opportunities for the deployer to control the model. But, of course, the user could have written some code to produce generations using released open-source model parameters.

with the system's interfaces.[41]

Forks in the supply chain There is a loop from generation back to the beginning of the supply chain. While not the most common contemporary practice, it is possible to use generations as training data for generative-AI models.[42] In this case, generation serves simultaneously as the creation of expressive works (i.e., stage 1) and data creation (i.e., stage 2) and generations can become inputs to dataset collection and curation processes (i.e., stage 3), which we indicate with an arrow in Figure 10.1. As we discuss in Section 10.3, this potential circularity also has implications for copyright.[43]

Alternatively, for the process of generation, some generative-AI systems interact with *external* deployed services, as is the case with ChatGPT plugins [451]. Such interactions between external services and generation further complicate the generative-AI supply chain that we depict in Figure 10.1. In particular, by

[41]These three options highlight additional observations about prompts. Thus far, we have primarily discussed generations as expressive works, but prompts could also be expressive works. The expressive example we gave above was: "`a big dog facing left wearing a spacesuit in a bleak lunar landscape with the earth rising in the background as an oil painting in the style of Paul Cezanne high-resolution aesthetic trending on artstation`". Sufficiently expressive prompts written by the direct user of a service could be subject to copyright. Context windows are so large, it is even possible for the user to prompt with an entire expressive work. As we discuss below in our copyright analysis, it is of course possible for this expressive work to have also been authored by another individual. Prompts could also be produced by generative AI, but this does not have the same authorship considerations. For example, Anthropic's team discussed using the entire text of *The Great Gatsby* as a prompt to demonstrate the long context window of their language model, Claude [27]. While *The Great Gatsby* is now in the public domain, it is easy to imagine another book entered as the prompt, or a copyrighted image as the prompt in an image-to-image system. Or copyrighted audio as input to an audio-to-audio model, etc. User-supplied prompts may be stored on system-deployers' servers for non-transient periods of time, and may even serve training data for a future model. Such prompts may also be used in model alignment (Section 10.2.8).

[42]Using model outputs as training data for future models has been a common practice in other settings. For instance, back-translation, the process of using a machine-translation model to generate additional training data (by translating data from one language to another) is a common technique [531].

[43]There are also concerns that this practice can have negative effects on model quality [539].

potentially integrating with other systems, the generation stage could implicate an entirely separate, unspecified number of supply chains consisting of entirely different organizations and actors. This, too, raises important copyright implications (what if news articles or short stories are integrated by the plugin?).

10.2.8 Model Alignment

The generative-AI supply chain does not stop with generation. As discussed above, model trainers try to improve models during both pre-training and fine-tuning. For pre-training, they monitor evaluation metrics, and may pause or restart the process to alter the datasets and algorithm used (Section 10.2.4); for fine-tuning, they continue training the base model with data that is specifically relevant for a particular task (Section 10.2.5). Both of these base model modifications are coarse: They make adjustments to the dataset and algorithm, and do not explicitly incorporate information into the model about whether specific generations are "good" or "bad," according to user preferences.[44]

There is a whole area of research, called **model alignment**, that attempts to meet this need [382].[45] The overarching aim of model alignment is to *align* model outputs with specific generation preferences (see Figure 10.1, stage 8). Currently, the most popular alignment technique is called **reinforcement learning with human feedback (RLHF)** [126, 455]. As the name suggests, RLHF combines collected human feedback data with a (reinforcement learning) algorithm in order to update the model. Human feedback data can take a variety of

[44]Of course, words like "good" and "bad" can have multiple valences, and resist the kind of quantification on which ML depends. See Lee et al. [351] (discussing the challenges of defining "good" and "bad" in the context of model behavior).

[45]See OpenAI [382] (for an introduction to InstructGPT, a model that is aligned with human feedback).

forms, which include user ratings of generations. For example, such ratings can be collected by including thumbs-up and thumbs-down buttons in the application user interface, which are intended to query feedback about the system's output generation. In turn, the reinforcement learning algorithm uses these ratings to adjust the model — to encourage more "thumbs-up" generations and fewer "thumbs-down" ones.[46]

Future training and alignment on the model may include both the inputted prompt and the generation in addition to the feedback provided. As discussed in the prior section, user-supplied prompts may include copyrighted content created by either the user themselves or by another party. Most generative-AI companies begin model alignment prior to deployment or release (Section 10.2.6). In this respect, model alignment complements other techniques, like input-prompt and output-generation filtering (Section 10.2.7).[47]

[46] In the reinforcement learning setting, data is not labeled as explicitly as it is in discriminative setting, e.g., our example of an image classifier, where each training data image has a label of either `cat` or `dog`. Instead, generations may be labeled "good" or "bad" based on human feedback, and the reinforcement learning algorithm updates the model in response to that feedback. In RLHF, feedback is generated by a person interacting with the system; however, RL can also use feedback automatically generated by an algorithm specification [32].

[47] Before making models publicly available, these companies contract with firms, like Scale AI [516], that simulate the user feedback process. These firms typically employ people to label generations as "good" or "bad," according to guidance from the generative-AI company. In general, the process of model alignment is a critical part of the supply chain. It serves as a mechanism for steering models away from generating potentially harmful outputs (See Cole [135], describing a book on mushroom foraging built from generations, which mistakenly indicate that toxic mushrooms are safe to eat) and toward the policies of the company or organization that deployed the model. See Google [394], OpenAI [454], Ganguli et al. [220] (documenting safety considerations, alignment, and RLHF at Google, OpenAI, and Anthropic).

10.3 Copyright and the Supply Chain

The hornbook statement of United States copyright doctrine is that original works of authorship are protected by copyright when they are fixed in a tangible medium of expression. A defendant directly infringes when they engage in conduct implicating one of several enumerated exclusive rights (reproducing, publicly distributing, etc.), with a work of their own that is substantially similar to a copyrighted work because it was copied from that work. Other parties may be held secondarily liable for conduct that bears a sufficiently close nexus to the infringement under one of several theories. Otherwise infringing conduct is legal when it is protected by one of several defenses, including the DMCA Section 512 safe harbors, fair use, or an express or implied license.

In this section, we first provide some brief background on what kinds of works copyright applies to (Section 10.3.1). We then apply aspects of the above orthodox, uncontested statement of copyright law to the generative-AI supply chain. We address issues of rights (Section 10.3.2), infringement (Sections 10.3.3 & 10.3.4), and fair use (Section 10.3.5). We defer discussion of safe harbors, licenses, paracopyright liability, and remedies to the longer version of our article [349]. Our goal is to be careful and systematic, not to say anything dramatically new.

10.3.1 What is copyrightable?

Copyright protects "(1) original works of authorship (2) fixed in any tangible medium of expression" [149].[48] "Original, as the term is used in copyright,

[48] 17 U.S.C. § 102(a) (numbering added).

means only that the work was independently created by the author (as opposed to copied from other works), and that it possesses at least some minimal degree of creativity" [202, p. 345] Fixation is satisfied when the work is embodied in a tangible object in a way that is "sufficiently permanent or stable to permit it to be perceived, reproduced, or otherwise communicated for a period of more than transitory duration" [152].[49]

We start with fixation. Unfixed works have no interaction with the generative-AI supply chain. A work must be fixed to be used as training data. Truly ephemeral creations, like unobserved dances and songs that are never recorded, will never be captured in a way that can be used as an input to a training algorithm. Datasets, models, applications, prompts, and generations are all fixed in computers and storage devices. Once it is fixed, however, any kind of original expression can be used as an input for generative AI.

The originality requirement distinguishes material that was created by a human author from facts that "do not owe their origin to an act of authorship" [202, p. 347]. In addition, some types of material are never copyrightable, including any "idea, procedure, process, system, method of operation, concept, [or] principle"[50] In practice, this means that the copyright in some works (e.g., product photographs) will be "thinner" and protect fewer aspects of the works than the "thicker" copyrights in others (e.g, abstract art), because the "range of creative choices that can be made in producing the works is narrow" [494, p. 1120]. In particular, any copyright in computer software — which is treated as a "literary work" for copyright purposes — typically excludes a great deal of functional material, such as coding conventions required by the choice of programming

[49] 17 U.S.C. § 101 (definition of "fixed").)
[50] 17 U.S.C. § 102(b).

language [514]. As a result, some individual training examples are uncopyrightable. (For example, birdsong-recognition AIs are trained on recordings of birds [305, 434].[51]) But other items are copyrightable, and those copyrights will be held by a variety of authors.

Training datasets will include different amounts and proportions of copyrighted material. A dataset of birdsong recordings will be almost entirely copyright-free, but a dataset of illustrations will contain numerous copyrighted works. Further, datasets *themselves* may be copyrightable as **compilations** [148],[52] "formed by the collection and assembling of preexisting materials or of data" [152].[53] A compilation is copyrightable as such when it features a sufficiently original "selection or arrangement" [202, p. 348]. Originality in selection is choosing *what to include* in the dataset; originality in arrangement is choosing *how to organize* it.

Generations raise a doctrinal question that has been debated for decades: who, if anyone, owns the copyright in the output of a computer program [512]? Although some have argued that the program itself should be regarded as the author, computer authorship is squarely foreclosed by U.S. copyright law [246]. So far, the courts have held firm to this line for AI generations. *Thaler* [574] upheld the Copyright Office's refusal to register copyright in an image allegedly "autonomously created by a computer algorithm running on a machine." The Copyright Office had held that the image lacked human authorship, and the court agreed.[54] The author of a generation — if anyone — is some human connected to the generation. The four immediately relevant possibilities are (1) au-

[51] See Kahl et al. [305]. Animals are not recognized as "authors" for copyright purposes. See *Naruto* [434].

[52] 17 U.S.C. § 103(a).

[53] 17 U.S.C. § 101 (definition of "compilation").

[54] That is, programs, like animals, are not "authors" within the meaning of the Copyright Act.

thor(s) whose works the model was trained on, (2) some entity in the generative-AI supply chain (e.g., the model trainer or fine-tuner; application developer), (3) the user who prompted a service for the specific generation, or (4) no one. As between these four possibilities, there is no one-size-fits-all answer. All four arise in actual generative-AI applications.

10.3.2 The Exclusive Rights

Copyright includes five relevant exclusive rights: reproduction, adaptation, public distribution, public performance, and public display.[55] Every stage in the generative-AI supply chain requires a reproduction and thus potentially implicates copyright. Because the remedies for infringement of a work are the same, regardless of whether the defendant violated one exclusive right or several, the precise dividing lines are often unimportant. We examine the adaptation right, and defer additional discussion to other work.

The adaptation right gives the copyright owner the exclusive right to "to prepare derivative works based upon the copyrighted work."[56] A derivative work combines the authorship in an existing (or "underlying") work with new authorship. In a compilation (Section 10.3.1), the underlying works are present in substantially unmodified form, whereas in a derivative work the underlying work is "recast, transformed, or adapted."[57]The adaptation right makes clear that copyright extends beyond literal similarity to incorporate changes of form, genre, and content such as translations, sequels, and film adaptations [230, 231, 513].

[55]17 U.S.C. § 106
[56]17 U.S.C. § 106(2)
[57]17 U.S.C. § 101 (definition of "derivative work).

A training dataset is probably not a derivative work of any of the works in it; it is more appropriately classified as a compilation "formed by the collection and assembling of preexisting materials" [152]. To the extent that a model is similar to a work it was trained on, it is a derivative work because it is "based on" its training data. (Section 10.3.3). Similarly, a prompt could be a reproduction or derivative of an existing work (as when a diffusion model is prompted with an image to infill) [27]. And generations are frequently derivative works of works in the training data or prompts, again subject to similarity.

10.3.3 Substantial Similarity

Substantial similarity is a qualitative, factual, and frustrating question. Two works are substantially similar when "the ordinary observer, unless he set out to detect the disparities, would be disposed to overlook them, and regard their aesthetic appeal as the same" [469, p. 489]. A common test is a "holistic, subjective comparison of the works to determine whether they are substantially similar in total concept and feel" [494, p. 1118]. This is not a standard that can be reduced to a simple formula that can easily be applied across different works and genres.[58] We discuss base models and generations below, and defer discussion of data, datasets, fine-tuned models, aligned models, and deployed services to other work.

[58]But see Scheffler et al. [518] (describing a principled computational basis for comparing works)

Pre-Trained/Base Models

A model is different in kind from the copyrightable works it was trained on. No viewer would say that the model has the same "total concept and feel" as a painting; no reader would say that it is substantially similar to a blog post; and so on. That said, the Copyright Act does not require that copies be directly human-intelligible to infringe. A Blu-Ray is not directly intelligible by humans, either, but it counts as a "copy" of the movie on it. Indeed, all digital copies are unintelligible. Instead, they are objects "from which the work can be perceived, reproduced, or otherwise communicated . . . *with the aid of a machine or device*" [152]. Thus, even if a model is uninterpretable, it might still be possible to "perceive[]" or "reproduce[]" a copyrighted work embedded in its parameters through suitable prompting. Indeed, there is substantial evidence that many models have memorized copyrighted materials [104, 109].[59] For example, Carlini et al. [104] shows how Stable Diffusion has memorized photographs.

A model might memorize more works or fewer [104, 105]. But at least some models memorize at least some works closely enough to pass the substantial-similarity test. On this view, a model is a substantially similar copy of a work when the model is capable of generating the work.[60] Note that this is direct infringement, not secondary (Section 10.3.4). The theory is not that the generation is an infringing copy, and that the model is a tool in causing that infringement in the way that a tape-duplicating machine might be a tool in making infringing

[59] See Carlini et al. [109] (GPT-2 memorizes training data); Carlini et al. [104] (Stable Diffusion and Imagen memorize images); Chang et al. [116] (suggestive evidence that GPT-4 memorizes training data).

[60] This is a sticky technical problem. Research has shown that memorization is not easily identifiable, and thus the amount of memorization in a model is not always or easily quantifiable. In particular, the choice of memorization identification technique and available information (e.g., knowledge of the training dataset, context window, etc.) affect the amount of memorization that can be identified. See, e.g., Carlini et al.[105].

(a) "an adventurous archaeologist with a whip and a fedora"

(b) "ice princess"

Figure 10.2: Generated by the authors using Midjourney.

cassettes [1]. Rather, the theory is that the model itself is an infringing copy, regardless of whether that particular generation is ever made.[61]

Generations

Some generations are nearly identical to a work in the model's training data (i.e., memorized). They are substantialy similar to that work. Other generations are very dissimilar from every work in the training data. There is no substantial similarity, because infringement is assessed on a work-by-work basis. Although it is in some sense based on all of the works in the training dataset, it does not infringe on any of them.[62] The hardest case is when an output is similar to a

[61] Alert readers will note the similarity to the debate over whether the mere act of making a work available without a download infringes the distribution right. See *London-Sire Records* [379]; see generally Menell [404].

[62] While it may be straightforward to pose the question: "is the given generation substantially similar to work 1," it is not at all straightforward to answer. Training datasets are massive. Manually comparing the generation to every single work in the dataset is infeasible; it would simply take too long. While automated methods could help identify works in the training set that are *likely to be* similar to the generation, there is no automated metric that can definitively say if two works are substantially similar. See generally Scheffler et al. [518] (which proposes one possibility for a metric for identifying substantial similarity)). Even with automated meth-

work in the training data in some ways, but dissimilar from it in other ways. This case is likely to arise in practice precisely because it lies in between the two extremes of memorized generations and original generations. Somewhere between them lies the murky frontier between infringing and non-infringing.

It is hard to make sweeping statements because of the factual intensity and aesthetic subjectivity of similarity judgments.[63] Whether a particular generation is substantially similar or not is ultimately a jury question requiring assessment of audiences' subjective responses to the works. Generative AI will produce cases requiring this lay assessment; it is impossible to anticipate in advance how lay juries will react to all of the possible variations. So, we will assume that lay audiences would say that some generations will infringe, but that it will not be possible to perfectly predict which ones.[64]

Even if complete answers are impossible, there are some interesting questions worth considering. As Matthew Sag observes [510], certain characters are so common in training datasets that models have "a latent concept [of them] that is readily identifiable and easily extracted." For example, prompting Midjourney and Stable Diffusion with `"snoopy"` produces recognizable images of Snoopy the cartoon beagle. Characters are a special case in copyright; some cases relax the rule that infringement is measured on a work-by-work basis, instead measuring the similarity of the defendant's character to one who appears

ods, checking *every* generation that a system produces against every other work in the training dataset to evaluate similarity is extremely computationally expensive.

[63] To quote Learned Hand on the idea-expression dichotomy, "Nobody has ever been able to fix that boundary, and nobody ever can" [444, p. 121].

[64] Notably, providing guarantees that any given generated work might not potentially infringe copyright is impossible if the training data contains copyrighted data. This is simply because provable guarantees require formal definitions, and there are no widely accepted formal definitions of substantial similarity. But see Scheffler et al. [518] (providing a possible starting point). Instead, current ML techniques focus on reducing the likelihood that generations from a model will closely resemble any of the model's training data.

in multiple works owned by the plaintiff.[65] But the "Snoopy effect" is not confined to characters. Some works are simply so prevalent in training datasets that models memorize them. As an uncopyrighted example, Van Gogh's *Starry Night* is easy to replicate using Midjourney; Sag's paper includes a replication of Banksy's *Girl with Balloon*. This looks like substantial similarity.

A variation of the Snoopy effect arises when a model learns an artist's recognizable *style*. ChatGPT can be prompted to write rhyming technical directions in the style of Dr. Seuss; DALL·E-2 can be prompted to generate photorealistic portraits of nonexistent people in the style of Dorothea Lange [113]. As with characters, these outputs have similarities that span a body of source works, even if they are not close to any one source work. The proper doctrinal treatment of style is a difficult question [552]. The Snoopy effect can also be triggered even without explicit prompting. The archaeologist example generated in Figure 10.2a features a dark-haired male character with stubble, wearing a brown jacket and white shirt, with a pouch slung across his shoulder. These are features associated with Indiana Jones, but neither the features nor `"indiana jones"` appear in the prompt. Some caselaw holds that these types of similarities are enough for infringement when the character is iconic enough [412].[66]

Other copyright doctrines, however, may limit infringement in Snoopy-effect cases. One of them is *scènes à faire*: creative elements that are common in a genre cannot serve as the basis of infringement. For example, [610, p. 50] explains that "drunks, prostitutes, vermin and derelict cars would appear in

[65] E.g., *DC Comics v. Towle* [164]; see generally Sag [510] (discussing caselaw and scholarship)

[66] See *Metro-Goldwyn-Mayer v. American Honda Motor Co.* [412] (car commercial featuring "a handsome hero who, along with a beautiful woman, lead a grotesque villain on a high-speed chase, the male appears calm and unruffled, there are hints of romance between the male and female, and the protagonists escape with the aid of intelligence and gadgetry" infringes on James Bond character).

any realistic work about the work of policemen in the South Bronx." Similarly, prompting Midjourney with "`ice princess`" produces portraits in shades of blue and white with flowing hair and ice crystals. (Figure 10.2b) Similarities to Elsa from *Frozen* arise simply because these are standard tropes of wintry glamour. Some of them may now be tropes *because* of the *Frozen* movies, but they are still uncopyrightable ideas, rather than protectable expression.[67]

To close this section, we note that not all similarity is infringing. Some similarities arise for innocent reasons. The defendant and the plaintiff might both have copied from a common predecessor work, and resemble each other because they both resemble the work they were based on. The similarities might consist entirely of accurate depictions of the same preexisting thing, like Grand Central Station at midday, and resemble each other because Grand Central Station resembles itself. The similarities might be purely coincidental. The plaintiff might even have copied from the defendant!

Copyright law therefore requires that the plaintiff prove that the defendant copied from their work, rather than basing it on some other source or creating it anew, an inquiry known as "copying in fact." This is a factual question. In some cases, there is direct evidence: e.g., the defendant admits copying or there is video of the defendant using tracing paper to copy a drawing. But in many cases, there are two kinds of indirect evidence: proof that the defendant had *access* to the plaintiff's work, and examples of "probative" *similarities* in the works themselves. Access shows that copying was possible, and similarities can rebut alternative innocent theories.[68]

[67] See *Nichols* [444, p. 121] ("Though the plaintiff discovered the vein, she could not keep it to herself; so defined, the theme was too generalized an abstraction from what she wrote. It was only a part of her 'ideas.' ")

[68] See generally *Skidmore* [544] (discussing proof of copying in fact); Latman [345] (distinguishing "probative" similarities that prove copying in fact from substantive similarities that consti-

10.3.4 Direct Infringement

We next discuss direct infringement and generations. We defer other supply-chain stages and analysis of indirect infringement to other work. Direct copyright liability has no mental element: it is "strict liability." All that is required is that they intentionally made the infringing copy. George Harrison's 1970 "My Sweet Lord" has the same melody and harmonic structure as the Chiffon's 1962 "He's so Fine"; the court held that "his subconscious knew it already had worked in a song his conscious mind did not remember," and found him liable for infringement [5, p. 180].

But direct copyright does have an element of "volitional conduct" [156]. Its purpose is to decide whether a defendant should be analyzed as a direct or indirect infringer.[69] Some courts have described the test in terms of causation: "who made this copy?"[70] The direct infringer is the party whose actions toward a specific item of content most proximately caused the infringing activity; anyone else is (potentially) an indirect infringer. Thus, for example, a service that can be used to upload and download infringing content that a user chooses does not engage in volitional conduct [467], but a service that curates a hand-picked selection of infringing content for users to download does [101].

The simplest case is where the same actor supplies both the model and the prompt.[71] Here, the subconscious-copying doctrine is a surprisingly good fit for AI generation. The model's internals are like the contents of George Harrison's brain: creatively effective, but not fully amenable to inspection. If I prompt

tute improper appropriation).

[69] See *Aereo* [23, 2512-13] at 2512-13 (Scalia, J., dissenting)

[70] See *Cartoon Network* [111, p. 130]; see also *Perfect 10, Inc. v. Giganews, Inc.* [467].

[71] Such as a text-to-image model developer using the model to create example prompt/generation pairs to display on their website.

an image model with "ice princess", I have set in motion a process that may draw on copyrighted works in the same way that George Harrison drew on other works he had heard. If that process generates Elsa, the resulting infringement is on me the same way that the infringement of "He's So Fine" was on Harrison. I could have taken greater care to check whether the image I was generating resembled a copyrighted work – just as George Harrison could have thought harder or asked more people whether the tune sounded familiar.

Matters are more complicated for generation services. Here, the question is whether the user and/or the provider should be treated as a direct infringer. There are at least three plausible answers, depending on the facts. First, the *user of the service* might be a direct infringer. If a user enters a prompt for "elsa and anna from frozen", the provider resembles a copy shop that provides a general-purpose tool and let users choose what to do with it [467]. Second, the *service provider* might be a direct infringer. If a user types in "heroic princesses" and the model generates a picture of Elsa and Anna, the user has acted innocently and it is the model that has narrowed down the space of possible outputs to one that happens to be infringing. Third, *both* the user of the service and service provider might be treated as direct infringers. Suppose the user inputs "frozen 3 screenplay" to a service that has been trained on thousands of Hollywood screenplays. Both the user and the service have the necessary volition to create a work that is substantially similar to the *Frozen* movies.

It seems unlikely, however, that a court would treat both service and user as indirect infringers. This would violate the doctrinal requirement that there be a direct infringer for indirect liability to attach, and it would leave both potentially responsible parties free of liability. The choice between the other three

cases is partly factual, and partly policy-driven. It is factual because there are clear paradigm cases in which the user of the service makes the choice for infringement, the service provider makes the choice for infringement, and the two conspire together to infringe. But it is policy-driven because, between these three poles, the identification of the direct infringer depends on which analogies one finds persuasive, and what one thinks copyright's goals are.[72]

10.3.5 Fair Use

Many stages of the generative-AI supply chain involve *prima facie* infringing reproductions, so copyright's all-purpose defense, fair use, will play a major role in making generative AI possible at all [150] Others have discussed the fair-use issues in detail [272, 429, 510, 551]. It is highly case-specific, so we will focus on only a few salient points. We discuss generations, taking each of the four fair-use factors in turn, and defer other stages to other work.

Factor One ("the purpose and character of the use ..." [150][73]): A use is transformative when "the quoted matter is used as raw material, transformed in the creation of new information, new aesthetics, new insights and understandings" [363, p. 1111]. The modification, remixing, and abstraction of input works literally involves exactly this kind of transformation. Some AI skeptics might deny that AI-generated material can be expressive. But as long as audiences find "new information, new aesthetics, new insights and understandings" in

[72]It is worth briefly noting that plugins could additionally pull in content from external sources, such as a news website, that gets included in a generation. Recall that this data is *not* included in training the model; instead, it is fed into the model at generation time to try to improve the quality of generations with more up-to-date information [451] Hypothetically, this content could get included verbatim in generations, leading to infringement issues in generation separate from those discussed above.

[73]17 U.S.C. § 107(1).

these generations, the goals of transformative use will be served.[74] Other generations will not be transformative. When a model outputs a memorized work, here is no transformation in content (Section 10.3.3). Other changes can also be non-transformative, e.g., memorized examples that are noisier than the source image. The noise is not new expression conveying new aesthetics. It is just noise. The rest of the first factor does not point one direction or the other. Generations can be put to commercial use (e.g., backgrounds for a music video) and to noncommercial use (e.g., illustrating an academic article on generative AI). Some outputs will be put to favored purposes like education and news reporting, while other outputs will be put to run-of-the-mill entertainment purposes.[75]

Factor Two ("the nature of the copyrighted work" [150][76]): This factor depends on the model in question. Some training data will be informational; some will be expressive. Most training data will have been "published" within the meaning of copyright law; otherwise, it would not be available as training data at all. A very small fraction of training data may be "unpublished" within the meaning of copyright law — i.e., it has been shared "(1) ... only to a select group (2) for a limited purpose and (3) with no right of further distribution by the recipients" [464, S. 6.31] — and included through express breach of confidence. Here, this factor will favor the plaintiff.

Factor Three ("the amount and substantiality of the portion used ..." [150][77]): This factor, like substantial similarity, will not systematically favor either side.

[74] See *Cariou* [103, p. 707] (focusing audience perceptions of works rather than author's intentions in assessing transformative use); see generally Heymann [275] (assessing transformative use from audience perspective); Liu [378] (discussing audience interests in copyright).

[75] See 17 U.S.C. § 107 [150] (favoring "purposes such as criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship, or research")

[76] 17 U.S.C. § 107(2)

[77] 17 U.S.C. § 107(3)

Some generations will closely resemble the works they were copied from; others will copy only small portions of the works.[78] Even for works that are transformative, it still matters whether the generation copies more than necessary. A "painting of a car driving in a snowstorm in the style of Frida Kahlo" might copy just Kahlo's brushwork or floral motifs, or it might also imitate the entire composition of one of her self-portraits.

Factor Four ("the effect of the use upon the potential market for ... the copyrighted work."[79]): The outputs of a non-generative AI do not compete in the market for a copyrighted work. These outputs could *reduce the demand* for the copyrighted work. For example, an AI-powered recommendation system might analyze the frames of a movie and assign it a low rating for visual interest. But the rating does not substitute for the movie in the market for movies. Viewers consume the rating to learn about movies, not to enjoy the expression in the rating. Any harm to the copyright owner is not fourth-factor harm [99]. The outputs of a generative-AI system, however, can substitute for a copyrighted work under the fourth factor. Consider the following variations on a theme:

- Instead of paying to obtain a copy of "The Old Sugarman Place" episode of *Bojack Horseman*, a user prompts a generative-AI system to generate `"'The Old Sugarman Place'"`. It generates a close duplicate — essentially a pirated edition at a lower price. This is a paradigmatic fourth-factor harm.
- The user prompts a generative-AI system to generate `"'The Old Sugarman Place'"`, and the system generates a non-exact copy with significant changes to the dialogue and animation. This episode, "The New

[78] See *Associated Press v. Meltwater U.S. Holdings, Inc.* [29] (rejecting fair use defense brought by news-monitoring service that reproduced substantial excerpts from articles for its customers)
[79] 17 U.S.C. § 107(4)

Sugarman Place," is also a direct competitor for this user's business. It might be a better or worse competitor, depending on how closely "The New Sugarman Place" matches "The Old Sugarman Place." But this is still factor-four harm.

- The user prompts a generative-AI system to generate a new episode of *Bojack Horseman*. The generation does not necessarily compete with "The Old Sugarman Place" itself.[80] Instead, it competes with commissioning the writers, animators, and voice cast to create new episodes, or with paying for a license to make new episodes.[81] This is also factor-four harm to the market for licenses and authorized derivatives. For example, in *Krofft* [540] McDonald's created advertisements in the unsettling style of the children's show *H.R. Pufnstuff*.

- An individual prompts a generative-AI system to produce a generation in a broad style, e.g., `"animated sitcom about depression"`. The output is a video with dialogue and animation that do not look much like *Bojack*. The output does not directly compete with "The Old Sugarman Place," or with any particular work or particular author. Instead, it competes with animated television in general. If the generative-AI system had not been available, the individual might have paid to watch *Bojack* or *Dr. Katz* or some other show. Many authors might view this as undercutting the market for their work. Here, the fourth factor is *not even relevant*, because the new video is not substantially similar to any existing work. If a human creative team made a new animated sitcom about depression, they would be celebrated for their creativity not sued for infringement.

[80] Perhaps the user has already watched all of the existing episodes.

[81] For another example, imagine that the user of a service prompts a text-to-image system to create a portrait of them in the style of a particular living artist; the generation is a substitute for commissioning the artist to paint one.

- An individual prompts a generative-AI system to produce a generation in a broad style, e.g. `"animated sitcom about depression"`. The output, however, is "The Old Sugarman Place." The difference between this and the first case is that the user does not know about the work that the generation substitutes for. This is a factor-four harm. The generative-AI system has diverted the individual from potentially learning about and paying to watch "The Old Sugarman Place."

To summarize, factors one, three, and four can point strongly in favor of fair use or strongly against, depending on the context, and factor two does not consistently point in either direction. We conclude that some generations will be fair uses and others will not.

10.4 Which Way from Here?

The generative-AI supply chain is extremely complex. So is copyright law. Putting the two of them together multiplies the intricacy. Two unsettling conclusions follow. First, because of the complexity of the *supply chain*, it is not possible to make accurate sweeping statements about the copyright legality of generative AI. Too much depends on the details of specific systems. All the pieces matter, from the curatorial choices in the training dataset, to the training algorithm, to the deployment environment, to the prompt supplied by the user. Courts will have to work through these details in numerous lawsuits and develop doctrines to distinguish among different systems and uses. Second, because of the complexity of *copyright law*, there is enormous play in the joints. Substantial similarity, fair use, and other doctrinal areas all have open-ended

tests that can reach different results depending on the facts a court emphasizes and the conclusions it draws. This complexity gives courts the flexibility to deal with variations in the supply chain. Paradoxically, it also gives courts the freedom to reach any of several different plausible conclusions about a generative-AI system. We explore some of the ways that courts might use their discretion to apply copyright law to generative AI (Section 10.4.1), and then discuss some of the considerations that courts should keep in mind (Section 10.4.2).

10.4.1 Possible Outcomes

There are a few boxes that courts may find it appealing to sort generative-AI systems into.

No Liability

First, courts might hold that neither services nor users are liable for copyright infringement. Under a combination of no substantial similarity and fair use, anything produced by a generative-AI system would be categorically legal. Models and services would also be legal because intermediate nonexpressive fair use would shield them. Training datasets would also usually be legal as well (except perhaps in cases of blatant infringement like Books3) [304, 319, 493]. They would be fair-use inputs to noninfringing downstream stages of the supply chain.

This regime is clear and simple. It would also be unstable. While this outcome might make sense for some generative-AI systems, it seems both unwork-

able for systems trained specifically to emulate the styles of particular creators, and retrieval systems that reproduce matching works exactly [73]. If all generative AI were categorically legal, then developers might start adding generative components to other systems in order to launder copyrighted works through them. The endpoint could be the effective collapse of copyright. Assuming that this is not an outcome that courts would willingly preside over, then, a blanket no-liability regime seems unlikely. Instead, courts would be more likely to find at least some infringement — so the question becomes where to draw the line.

Liability for Generations Only

Second, courts could draw a line between services and users. In this regime, only generations would be treated as infringing.[82] In this world, generative-AI systems would be creative tools like Photoshop.[83] The user would be responsible for making sure that anything they create with the tools is noninfringing, but the tools would be shielded under something like a strong *Sony* rule, assembled out of a combination of no substantial similarity, no indirect infringement, and/or fair use. This result might be unfair to users whose infringements resulted from systems producing generations that reproduce material in the underlying model's training dataset, through no choice or fault of their own. But this is arguably the same kind of situation that some courts currently countenance when they hold that users can be liable for embedding images from Instagram even though Instagram is not liable for hosting those images [541].

The main difficulty with this regime would be policing against systems de-

[82]Here, we use the term "user" broadly. A user could be a customer using a web application to produce a generation, a developer using an API to produce a generation in their own code, a developer using an API to produce a generation for a company, etc.

[83]Sometimes literally so. See Adobe [11].

signed specifically for infringement. Something like the *Grokster* [411] rule, carefully followed, might suffice. The providers of a service that was geared to produce infringing outputs could be held liable. So could the publishers or deployers of a model that had been trained or fine-tuned to optimize its effectiveness at infringement. So could the curator of a dataset that included only infringing works, or was intentionally organized to meet the needs of a model known to be intentionally trained for infringement. At every stage, a party would be held responsible only for its own actions directed towards increasing the use of a system for infringement.

Notice and Removal

Courts could treat generative-AI services as generally legal, but require them to respond to knowledge of specific infringements under a *Napster*-like rule [2]. One plausible route to this regime would be to treat infringing generations as creating direct liability for users and only indirect liability for service providers. Another would use fair use to shield service providers as long as they took reasonable overall precautions, including responding when they had sufficient knowledge of infringement. And a third would be to find liability but craft an injunction that only required services to act against infringement they were aware of.[84]

[84]Regardless of which of these doctrinal routes a court took, there would be an inevitable gravitational force pulling the provider's duties towards the duties of a service provider under Section 512(c) or (d). This is not because Section 512 applies to generative-AI services. It largely does not — analysis that we defer to other work. Instead, the Section 512 doctrines may be a convergence point because courts have now had two decades of experience — which means two decades of precedents — with the Section 512 safe harbors. These precedents have come to set expectations — among copyright owners, in the technology industry, in the copyright bar, and in the judiciary — for what legally "responsible" behavior by an online intermediary looks like. A generative-AI service operator that does not appear to be making a good-faith effort to achieve something like this system may strike a court as intending to induce infringement, not making a good-faith effort to comply with an injunction, etc.

If courts end up recreating a notice-and-takedown regime, they would likely settle on familiar elements from the DMCA notice-and-takedown provision of Section 512: a way for copyright owners to give notice of infringement, block infringing generations on notice, block infringing generations on actual knowledge, block infringing generations on red-flag knowledge, avoid having a business model that directly ties income to infringement, and terminate the abilities of repeat infringers to continue making generations.

This is a very difficult technical problem. It would be much harder for a generative-AI system to implement than it is for a hosting platform to implement Section 512 compliance. The reason is that a notice directed to a hosting provider under Section 512(c) must include "Identification of the material that is claimed to be infringing ...and information reasonably sufficient to permit the service provider to locate the material" [153].[85] A valid notice is a roadmap; it tells the hosting provider exactly what to take down to comply. That material already exists, and the hosting provider can compare it to the copyrighted work to verify that they are substantially similar. But a notice to a generative-AI system is a notice against future generations, which may be different from each other and resemble the copyrighted work in different ways. Filtering for this kind of much more inexact match is much harder technically.[86] Further, there is no simple analogue for takedown in generative-AI models. Removing the

[85]U.S.C. § 512(c)(3)(A)(iii).

[86]That said, matching material against a catalog of copyrighted works is a problem that has been very approximately solved by major social networks, which use perceptual hashing to prevent the upload of various kinds of identified content. Generative-AI companies could at least add similar perceptual-hash-driven filtering to the outputs of their models, but clearly this would only solve part of the problem [294, 352]. The challenges of implementing removal for models are even harder. A service can add filters on the input and output sides — monitoring prompts and scanning outputs. It can also fine-tune or align the model, or provide it with an overall prompt that instructs the model to respond in ways that reduce its propensity to infringe. Further, a model by itself does not implement these controls. The model cannot control how it is prompted or what the user does with the output. The model cannot stop anyone from fine-tuning it to remove its guardrails.

influence of a particular example on a model is an active and unsolved area of research [79, 405].[87]

Infringing Models

A fourth possibility is that some or all generative-AI services are illegal because models themselves infringe. This outcome is an existential threat to model trainers and service providers; it makes their operations *per se* copyright infringement. It is also the outcome being sought by the class-action plaintiffs in high-profile lawsuits against OpenAI, Stability AI, and some of their partners. In this regime, the most important component of copyright law would become licensing. Models could only be trained on data that had been licensed from the copyright owners; the terms under which those models and their generations could be used would have to be negotiated as part of the licensing agreement.[88]

10.4.2 Lessons

Having discussed what courts and policymakers could do, we now consider what they should do. In keeping with our bottom line — *the generative-AI supply chain is too complicated to make sweeping rules prematurely* — we offer a few general observations about the overall shape of copyright and generative AI that courts and policymakers should keep in mind as they proceed.

[87] Absent the ability to do so, the safest bet is to retrain the model from scratch. Due to the time and expense required to retrain a model, it will often be infeasible to retrain it simply to remove infringing works, and completely unworkable to retrain on each new notice. We defer further discussion of how courts could deal with this difficulty to other work.

[88] Each model would have a fully licensed training dataset, and the question of infringement would not arise except in cases where there were infringing works in the dataset itself or some other failure of quality control somewhere along the supply chain.

First, *copyright touches every part of the generative-AI supply chain*. Every stage from training data to alignment can make use of copyrighted works. Generative AI raises many other legal issues: Can a generative-AI system commit defamation [37, 91, 224, 270, 603]? Can a generative-AI system do legal work [123] and should they be allowed to [396]? But these issues pertain to outputs of a generative-AI system–copyright pervades every step of the process.

Second, *copyright concerns cannot be localized* to a single link in the supply chain. Decisions made by one actor can affect the copyright liability of another actor far away in the supply chain. Whether an output looks like Snoopy or like a generic beagle depends on what images were collected in a dataset, which model architecture and training algorithms are used, how trained models are fine-tuned and aligned, how models are embedded in deployed services, what the user prompts with, etc. Every single one of these steps could be under the control of a different person.

Third, *design choices matter*. There are obvious choices about copyright, like whether to train on unlicensed data (which can affect downstream risks), and how to respond to notices that a system is producing infringing outputs (which can affect upstream risks). But subtler architectural choices matter, too. Different settings on a training algorithm can affect how much the resulting model will memorize specific works. Different deployment environments can affect whether users have enough control over a prompt to steer a system towards infringing outputs. Copyright law will have to engage with these choices — as will AI policy.

Fourth, *fair use is not a silver bullet*. For a time, it seemed that training and using AI models would often constitute fair use. In such a world, AI devel-

opment is generally a low-risk activity, at least from a copyright perspective. Yes, training datasets and models and systems may all include large quantities of copyrighted works — but they will never be shown to users. Generative AI scrambles this assumption. The serious possibility that some generations will infringe means that the fair-use analysis at every previous stage of the supply chain is up for grabs again.

Fifth, *the ordinary business of copyright law still matters*. Courts will need to make old-fashioned, retail judgments about individual works — e.g., how much does this image resemble Elsa in particular, rather than generic tropes of fantasy princesses? Courts *must* leave themselves room to continue making these retail judgments on a case-by-case basis, responding to the specific facts before them, just as they always have. Perhaps eventually as society comes to understand what uses generative AI can be put to and with what consequences, it will reconsider the very fundamentals of copyright law. But until that day, we must live with the copyright system we have. And that system cannot function unless courts are able to say that some generative-AI systems and generations infringe, and others do not.

Sixth, *analogies can be misleading*. There are plenty of analogies for generative AI ready to hand. A generative-AI model or system is like a search engine, or like a website, or like a library, or like an author, or like any number of other people and things that copyright has a well-developed framework for dealing with. These analogies are useful, but we wish to warn against treating any of them as definitive. As we have seen, generative AI is and can consist of many things. It is also literally a generative technology: it can be put to an amazingly wide variety of uses [648]. And one of the things about generative technologies

is that they cause convergence [433]. precisely because they can emulate many other technologies, they blur the boundaries between things that were formerly distinct. Generative AI can be like a search engine, and also like a website, a library, an author, and so on. Prematurely accepting one of these analogies to the exclusion of the others would mean ignoring numerous relevant similarities — precisely the opposite of what good analogical reasoning is supposed to do.

10.5 Conclusion

Our conclusion is simple. "Does generative AI infringe copyright?" is not a question that has a yes-or-no answer. There is currently no blanket rule that determines which participants in the generative-AI supply chain are copyright infringers. The underlying systems are too diverse to be treated identically, and copyright law has too many open decision points to provide clear answers. Copyright is not the only, or the best, or the most important way of confronting the policy challenges that generative AI poses. But copyright is here, and it is asking good questions about how generative-AI systems are created, how they work, how they are used, and how they are updated. These questions deserve good answers, or failing that, the best answers our copyright system is equipped to give.

CHAPTER 11

CONCLUSION

This dissertation puts forth a vision for a new research field that studies problems at the intersection of machine learning, law, and policy. At its center are questions that concern the challenges for taking reliability seriously across machine learning: the importance of careful, meaningful metric design, methodologies for making sure that we can measure these metrics efficiently and dependably at scale, and successful communication with legal scholars and policymakers about what measurements can (and cannot) tell us about the capabilities and risks of ML systems. In service of this vision, this dissertation covers research contributions in three different, yet cross-cutting, themes: sources of arbitrariness in machine learning (Part I), taming randomness in scalable, reliable ML algorithms (Part II), and evaluating generative-AI systems (Part III).

In Part I, we describe different ways that non-determinism can bring about arbitrary outcomes in ML experiments. We quantify and mitigate two particular sources of machine-learning-related arbitrariness: (1) arbitrariness that can result in conclusions from non-deterministic, human-made decisions in experiments that involve hyperparameter optimization (Chapter 2), and (2) arbitrariness that can result in social prediction contexts when we do not account for variance in possible learned decision rules, due to randomness in the training process (Chapter 3).

Arbitrariness is especially relevant in the law. One of the goals of legal rules is to remove arbitrariness in decision-making, so that the law is predictable, consistent, and fair — qualities that are essential for due process [218, 570]. We make important, novel connections between non-determinism-induced ar-

bitrariness in ML and arbitrariness in the law. Our work in this area shows why ML-related arbitrariness is meaningfully different from other types of arbitrariness that the law considers, which has important consequences for how law- and policymakers should reason about and regulate the deployment of ML systems in public contexts.

In Part II, we detail algorithmic contributions in scalable uncertainty estimation and distributed optimization. This research contends with improving scalability without sacrificing reliability. For the former, reliability involves guaranteeing that our Bayesian inference algorithms converge to the true posterior distribution, so that we can get accurate estimates of model uncertainty (Chapter 5). For the latter, we prove that there are better-than-random example orders for SGD-based distributed optimization; our algorithm converges provably faster than random example ordering, and also exhibits better generalization in practice (Chapter 6).

Both of these chapters harness randomness in sampling and optimization, respectively, to construct more scalable/efficient, reliable algorithms. In computing, scalabilty/efficiency and reliability are often in trade-off; it is challenging to achieve greater scalability or efficiency without sacrificing reliability. In both of these chapters, we leverage low-level technical details about sampling and optimization, respectively, so that we can relax these trade-offs.

We then discuss how analogous trade-offs are very common in law and policy contexts. Policymakers trade-off between reliability and scalability/efficiency all the time, in domains as complex as federal risk assessment and health policy. Understanding these trade-offs is often sufficient for making sense of underlying implementation decisions, which impact overall algorithm and sys-

tem behavior. As a result, we make the case that such trade-offs are a useful level of abstraction for communicating with policymakers and other non-expert stakeholders about the capabilities and risks of ML systems (Chapter 7).

Part III also involves algorithms in scalable machine learning, but makes its central contributions in evaluating generative-AI system. We put forth methodology for feasible measurement of memorization at contemporary-LLM scale. Since we do not know the training datasets of most state-of-the-art LLMs, we develop a proxy for a training dataset that we use to validate memorization. In doing so, we reveal interesting patterns in memorization across open and semi-closed models [435]. A variant of our attack, which gets the aligned ChatGPT to diverge from its chatbot-style outputs, shows that ChatGPT memorizes orders of magnitude more than was previously known, and much more than any other model that we tested. This work represents the first successful, large-scale memorization attack on an aligned, deployed, closed production system (Chapter 8).

One of the key concerns around regurgitating memorized training data is that these generations can contain verbatim copies of content that is possibly under copyright. Such copyright concerns suggest a natural alternative: training generative-AI models on public domain or explicitly licensed curated data. We explore this in depth for text-to-image latent diffusion models: we curate a training dataset of Creative Commons licensed images, for which we produce synthetic captions, and we use this image-caption dataset to train variants of the Stable Diffusion 2 architecture. This work involves a variety of ML-systems contributions, as well as a starting place for showing the benefits and limitations of training on licensed data (Chapter 9).

Both of these chapters provide a great intuition for why generative AI presents significant challenges for U.S. copyright law. However, since their main contributions are in machine learning, they do not go into detail about their relationship with copyright. We conclude this part by closing this loop. We present an abridged version of our landmark article on copyright and the generative-AI supply chain, which defines a rich framework for reasoning precisely about the many ways that copyright law presents significant issues for generative AI, and how generative AI, in turn, presents novel, significant issues for copyright law (Chapter 10).

By making contributions in these three interrelated themes, this dissertation demonstrates that research on reliable measurement for ML is intrinsically tied with research in law and policy. While these are different disciplines, they are actually two complementary sides of the same research vision. They serve the same goals of trying to rigorously define what it means to be "reliable" and how to implement reliability in practice. This requires rigorous knowledge about the capabilities and risk of ML models (and the systems in which they are embedded). Developing such knowledge depends on fundamental algorithmic and methodological contributions in machine learning. It also requires meaningful engagement with concrete law and policy questions about what we want ML systems to do in the world.

There is a virtuous cycle between these two sides: research on reliable measurement for machine learning has direct implications for law and policy, and work in law and policy raises novel questions to tackle concerning how to define and conduct reliable measurement for machine learning. I have been very fortunate throughout my Ph.D. to be able to engage deeply with both sides of

this research vision — to ask and answer questions that requires developing insights in machine learning, law, and policy.

This dissertation provides a tour of some of scholarship that has developed these insights. However, it does not discuss the work I have done to directly engage with policymakers in practice. Most of this work has occurred through an organization called The Center for Generative AI, Law, and Policy Research (The GenLaw Center), which I co-founded with my co-authors, Katherine Lee and James Grimmelmann. This organization originated through an ICML '23 workshop that we co-organized, and has grown into and incubated a new research community and field. We have had a significant impact on the trajectory of some Ph.D. students' doctoral work, and have an ongoing impact on U.S. AI policy.

Most recently, we held a workshop in Washington, D.C., which we co-organized with Carnegie Mellon University, the Georgetown Law Center, and the Center for Democracy and Technology. This workshop centered on educating policymakers about generative-AI evaluations, and what we can and cannot measure about generative-AI systems — in other words, the good, the bad, and the hype. The work that we are doing through The GenLaw Center — both our policy outreach and the research we are conducting — is exactly in line with some of the action items called for in President Biden's executive order [576], among other policy mandates.

The great news is, we have already been doing this work together for a while. We have been making, and continue to make, fundamental contributions in research and practice at the intersection of machine learning, law, and policy. It is a bit surreal to see some of the work in this dissertation become nationally,

or even internationally relevant. But this also means it is an especially exciting time to continue my research vision, beyond the introduction that has been presented in these chapters.

Part IV

Appendix

APPENDIX A

COMPREHENSIVE LIST OF PH.D. WRITING

*Equal contribution

27. **A. Feder Cooper*** and James Grimmelmann*. "The Files are in the Computer: Copyright, Memorization, and Generative AI." Forthcoming, *Chicago-Kent Law Review*. 2024 [139].

26. Nicholas Carlini, Daniel Paleka, Krishnamurthy Dj Dvijotham, Thomas Steinke, Jonathan Hayase, **A. Feder Cooper**, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, Eric Wallace, David Rolnick, Florian Tramèr. "Stealing Part of a Production Language Model." *International Conference on Machine Learning 2024* (*ICML '24*) [108].

25. Daniel McDuff, Tim Korjakow, Scott Cambo, Jesse Josua Benjamin, Jenny Lee, Yacine Jernite, Carlos Muñoz Ferrandis, Aaron Gokaslan, Alek Tarkowski, Joseph Lindley, **A. Feder Cooper**, and Danish Contractor. "On the Standardization of Behavioral Use Clauses and Their Adoption for Responsible Licensing of AI." *International Conference on Machine Learning 2024* (*ICML '24*) [399].

24. **A. Feder Cooper***, Katherine Lee*, and James Grimmelmann*. "Talkin' 'Bout AI Generation: Copyright and the Generative-AI Supply Chain." Forthcoming, *Journal of the Copyright Society*. 2024 [349].

23. **A. Feder Cooper***, Katherine Lee*, and James Grimmelmann*. "Talkin' 'Bout AI Generation: Copyright and the Generative-AI Supply Chain (The Short Version)." *The 3rd ACM Symposium on Computer Science and Law (CSLAW '24)*. 2024. **Long Presentation** [350].

22. Aaron Gokaslan, **A. Feder Cooper**, Jasmine Collins, Landan Seguin, Austin Jacobson, Mihir Patel, Jonathan Frankle, Cory Stephenson, and Volodymyr Kuleshov. "CommonCanvas: Open Diffusion Models Trained on Creative-Commons Images." *Conference on Computer Vision and Pattern Recognition 2024* (*CVPR '24*). 2024 [236].

21. **A. Feder Cooper**, Katherine Lee, Madiha Zahrah Choksi, Solon Barocas, Christopher De Sa, James Grimmelmann, Jon Kleinberg, Siddhartha Sen, and Baobao Zhang. "Arbitrariness and Social Prediction: The Confounding Role of Variance in Fair Classification." *38th AAAI Conference on Artificial Intelligence* (*AAAI '24*). 2024. **Best Paper Honorable Mention** [141].

20. Milad Nasr*, Nicholas Carlini*, Jonathan Hayase, Matthew Jagielski, **A. Feder Cooper**, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. "Scalable Extraction of Training Data from (Production) Language Models." 2023. Under submission [435].

19. **A. Feder Cooper***, Wentao Guo*, Khiem Pham*, Tiancheng Yuan, Charlie F. Ruan, Yucheng Lu, and Christopher De Sa. "Coordinating Distributed Example Orders for Provably Accelerated Training." *Conference on Neural Information Processing Systems 36 (NeurIPS '23)*. 2023 [140].

18. Kweku Kwegyir-Aggrey, **A. Feder Cooper**, Jessica Dai, John Dickerson, Keegan Hines, and Suresh Venkatasubramanian. "Repairing Regressors for Fair Classification at Any Decision Threshold." *Workshop on Algorithmic Fairness through the Lens of Time* at *NeurIPS 2023*. 2023. **Oral** [336].

17. **A. Feder Cooper***, Katherine Lee*, James Grimmelmann*, Daphne Ippolito*, Christopher Callison-Burch, Christopher A. Choquette-Choo, Niloofar Mireshghallah, Miles Brundage, David Mimno, Madiha Zahrah

Choksi, Jack M. Balkin, Nicholas Carlini, Christopher De Sa, Jonathan Frankle, Deep Ganguli, Bryant Gipson, Andres Guadamuz, Swee Leng Harris, Abigail Z. Jacobs, Elizabeth Joh, Gautam Kamath, Mark Lemley, Cass Matthews, Christine McLeavey, Corynne McSherry, Milad Nasr, Paul Ohm, Adam Roberts, Tom Rubin, Pamela Samuelson, Ludwig Schubert, Kristen Vaccaro, Luis Villa, Felix Wu, and Elana Zeide. "Report of the 1st Workshop on Generative AI and Law." 2023. Technical Report [142].

16. **A. Feder Cooper***, Katherine Lee*, James Grimmelmann, and Daphne Ippolito. "AI and Law: The Next Generation (An explainer series)." 2023. Technical Report [351].

15. **A. Feder Cooper**, Katherine Lee, Madiha Zahrah Choksi, Solon Barocas, Christopher De Sa, James Grimmelmann, Jon Kleinberg, Siddhartha Sen, and Baobao Zhang. "Distribution Justice: Variance, Uncertainty, and Rules in Machine Learning and Law." *Privacy Law Scholars Conference*. 2023.

14. **A. Feder Cooper**, Jonathan Frankle, and Christopher De Sa. "Non-Determinism and the Lawlessness of Machine Learning Code." *The 2nd ACM Symposium on Computer Science and Law (CSLAW '22)*. 2022. **Long Presentation** [138].

13. **A. Feder Cooper** and Karen Levy. "Fast or Accurate? Governing Conflicting Goals in Highly Autonomous Vehicles." *Colorado Technology Law Journal*, Vol. 20. 2022 [143].

12. **A. Feder Cooper**, Solon Barocas, Karen Levy, and Gili Vidan. " 'We have met the enemy and it is us': Debating the ethics of computing in the pages of *CACM*." *2022 Workshop of the The Special Interest Group for Computing, Information, and Society (SIGCIS '22)*. 2022.

11. **A. Feder Cooper***, Emanuel Moss*, Benjamin Laufer, and Helen Nissenbaum. "Accountability in an Algorithmic Society: Relationality, Responsibility, and Robustness in Machine Learning." *Proceedings of the 5th ACM Conference on Fairness, Accountability, and Transparency (FAccT '22)*. 2022 [146].

10. **A. Feder Cooper** and Gili Vidan. "Making the Unaccountable Internet: The Changing Meaning of Accounting in the Early ARPANET." *Proceedings of the 5th ACM Conference on Fairness, Accountability, and Transparency (FAccT '22)*. 2022 [147].

9. Benjamin Laufer, **A. Feder Cooper***, Sameer Jain*, Jon Kleinberg, and Hoda Heidari. "Four Years of FAccT: A Reflexive, Mixed-Methods Analysis of Research Contributions, Shortcomings, and Future Prospects." *Proceedings of the 5th ACM Conference on Fairness, Accountability, and Transparency (FAccT '22)*. 2022 [346].

8. **A. Feder Cooper**, Yucheng Lu, Jessica Zosa Forde, and Christopher De Sa. "Hyperparameter Optimization Is Deceiving Us, and How to Stop It." *Conference on Neural Information Processing Systems 34 (NeurIPS '21)*. 2021 [145].

7. **A. Feder Cooper**, Karen Levy, and Christopher De Sa. "Accuracy-Efficiency Trade-Offs and Accountability in Distributed ML Systems." *Proceedings of the 2021 ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization (EAAMO '21)*. 2021. **Oral** [144].

6. **A. Feder Cooper**, Maria Antoniak, Christopher De Sa, Marilyn Migiel, and David Mimno. " '*Tecnologica cosa*': Modeling Storyteller Personalities in Boccaccio's *Decameron*." *SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature* at *The 2021*

Conference on Empirical Methods in Natural Language Processing (EMNLP '21). 2021 [137].

5. **A. Feder Cooper** and Ellen Abrams. "Emergent Unfairness in Algorithmic Fairness-Accuracy Trade-Off Research." *Proceedings of the 2021 AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society (AIES '21)*. 2021. **Oral** [136].

4. **A. Feder Cooper***, Jessica Zosa Forde*, Kweku Kwegyir-Aggrey, Christopher De Sa, and Michael Littman. "Model Selection's Disparate Impact in Real-World Deep Learning Applications." *Workshop on the Science and Engineering of Deep Learning* at *The Ninth International Conference on Learning Representations (ICLR '21)*. 2021. **Oral** [210].

3. **A. Feder Cooper***, Ruqi Zhang*, and Christopher De Sa. "Asymptotically Optimal Exact Minibatch Metropolis-Hastings." *Conference on Neural Information Processing Systems 33 (NeurIPS '20)*. 2020. **Spotlight** [641].

2. Ruqi Zhang, **A. Feder Cooper**, and Christopher De Sa. "AMAGOLD: Amortized Metropolis Adjustment for Efficient Stochastic Gradient MCMC." *Proceedings of the Twenty-third International Conference on Artificial Intelligence and Statistics (AISTATS '20)*. 2020 [642].

1. **A. Feder Cooper**. "Imperfection is the Norm: A Computer Systems Perspective on IoT and Enforcement." *(Im)Perfect Enforcement Conference*, Information Society Project at Yale Law School. 2019. **Plenary session**.

APPENDIX B

APPENDIX FOR HYPERPARAMETER DECEPTION

Term	Explanation	Example
HPO	Acronym for hyperparameter optimization	
$\mathcal{J}, \mathcal{K}$	Used as examples of arbitrary optimizers	
P	Arbitrary atomic proposition	
p, q, ϕ	Used as arbitrary or (when specified) specific logical formulas	p = "Non-adaptive optimizers have higher test accuracy than adaptive optimizers."
HP(s)	Acronym for hyperparameter(s)	
$\ell \in \mathcal{L}$,	Log (Definition 1); log set	Figure B.4; Log for running HPO using SGD
T	The total time it took to run HPO to produce a log ℓ	
$\mathcal{I}$	A set of integers	Typically the 64-bit integers
r	Random seed; $r \in \mathcal{I}$	
G	Pseudo-random number generator; $G(r)$; $G : \mathcal{I} \to \mathcal{I}^{\infty}$	
PRNG	Acronym for pseudo-random number generator; G	
H	HPO procedure (Definition 2)	SGD, VGG-16 grid search experiment
H_*	A randomized algorithm used in H	Random search
$c \in C$	Hyper-HP configuration; of set of allowable such configurations for H_*	powers-of-2 grid spacing; configurations the demon has access to
$\lambda \in \Lambda$; λ^*	HP config. used to run an HPO pass; of allowable HP configs., determined by c; λ^* is the output HP config. that performs the best	$\alpha = 1$ in Wilson; allowable α values, e.g. [.001, .01, 1]
$\mathcal{A}$; $\mathcal{A}_\lambda$	Training algorithm; parameterized by HPs λ	SGD; SGD with $\alpha = 1$
$\mathcal{M}$; $\mathcal{M}_\lambda$	Model; parameterized by HPs λ	VGG16
X	A dataset	CIFAR-10
α	Learning rate	Figure B.1, $\alpha = 1$
ϵ	Adam-specific HP	Figure B.1, we set $\epsilon = 10^{12}$
EHPO	Epistemic HPO (Definition 3)	Our defended random search in Section 2.5
$\mathcal{H}$	Set of HPO procedures H	
$\mathcal{P}$	Set of concluded logical formulas; $p \in \mathcal{P}$	
$\mathcal{F}$	A function that maps a set of HPO logs $\mathcal{L}$ to a set of logical formulas $\mathcal{P}$	$\mathcal{F}_*$ (skeptical belief function); $\mathcal{F}_{\text{n}}$ (naive belief function)
$\Box$	Modal logic operator for "necessary"	$\Box p$ reads "It is necessary that p
$\Diamond$	Modal logic operator for "possible"	$\Diamond p$ reads "It is possible that p
$\vdash$	Indicates a theorem of propositional logic	$\vdash Q \to \Box Q$ (necessitation)
$\Diamond_t$	EHPO modal operator (Section 2.4.2; Definition 5)	
$\mathcal{B}$	Belief modal operator	$\mathcal{B}_*$ (skeptical belief); $\mathcal{B}_{\text{n}}$ (naive belief)
$\sigma \in \Sigma$	A randomized strategy function that specifies EHPO actions; set of all such strategies (Section 2.4.2, Definition 4)	
$\sigma(\mathcal{L})$	Distribution over concrete actions for log set	
$\sigma[\mathcal{L}]$	The logs output from running σ on $\mathcal{L}$	
$\tau_\sigma(\mathcal{L})$	Total time spent executing $\sigma[\mathcal{L}]$	
$\models$	Denotes "models"	$\mathcal{L} \models \Diamond_t p$: $\mathcal{L}$ model that p is possible in t
γ	Renyi-∞-divergence constant upper bound (Theorem 1)	
K, R	Numbers of independent random search trials (Section 2.5)	
κ	Subsampling size (Algorithm 1)	We set $\kappa = 11$ (Section 2.5)
M	Subsampling budget (Algorithm 1)	We set $M = 10000$ (Section 2.5)
δ	Skeptical reasoner conclusion threshold (Algorithm 1)	See Table 2.1

B.1 Definitions Reference

Hyper-hyperparameters (hyper-HPs) are HPO-procedure-input values, such as the spacing between different points in the grid for grid search and the distributions to sample from in random search.

Definition 1. *A log ℓ records all the choices and measurements made during an HPO run, including the total time T it took to run. It has all necessary information to make the HPO run reproducible.*

Definition 2. *An HPO procedure H is a tuple $(H_*, C, \Lambda, \mathcal{A}, \mathcal{M}, G, X)$ where H_* is a randomized algorithm, C is a set of allowable hyper-HPs (i.e., allowable configurations for H_*), Λ is a set of allowable HPs (i.e., of HP sets λ), $\mathcal{A}$ is a training algorithm (e.g. SGD), $\mathcal{M}$ is a model (e.g. VGG16), G is a PRNG, and X is some dataset (usually split into train and validation sets). When run, H_* takes as input a hyper-HP configuration $c \in C$ and a random seed $r \in \mathcal{I}$, then proceeds to run $\mathcal{A}_\lambda$ (on $\mathcal{M}_\lambda$ using $G(r)$ and data[1] from X) some number of times for different HPs $\lambda \in \Lambda$. Finally, H_* outputs a tuple (λ^*, ℓ), where λ^* is the HP configuration chosen by HPO and ℓ is the log documenting the run.*

Definition 3. *An **epistemic hyperparameter optimization procedure (EHPO)** is a tuple $(\mathcal{H}, \mathcal{F})$ where $\mathcal{H}$ is a set of HPO procedures H (Definition 2) and $\mathcal{F}$ is a function that maps a set of HPO logs $\mathcal{L}$ (Definition 1) to a set of logical formulas $\mathcal{P}$, i.e. $\mathcal{F}(\mathcal{L}) = \mathcal{P}$. An execution of EHPO involves running each $H \in \mathcal{H}$ some number of times (each run produces a log ℓ) and then evaluating $\mathcal{F}$ on the set of logs $\mathcal{L}$ produced in order to output the conclusions $\mathcal{F}(\mathcal{L})$ we draw from all of the HPO runs.*

[1] Definition 2 does not preclude cross-validation, as this can be part of H_*. The input dataset X can be split in various ways, as a function of the random seed r.

Definition 4. *A randomized* ***strategy*** $\boldsymbol{\sigma}$ *is a function that specifies which action the demon will take. Given* $\mathcal{L}$*, its current set of logs,* $\boldsymbol{\sigma(\mathcal{L})}$ *gives a distribution over concrete actions, where each action is either 1) running a new* H *with its choice of hyper-HPs* c *and seed* r *2) erasing some logs, or 3) returning. We let* Σ *denote the set of all such strategies.*

Definition 5. *Let* $\boldsymbol{\sigma[\mathcal{L}]}$ *denote the logs output from executing strategy* σ *on logs* $\mathcal{L}$*, and let* $\boldsymbol{\tau_\sigma(\mathcal{L})}$ *denote the total time spent during execution.* $\tau_\sigma(\mathcal{L})$ *is equivalent to the sum of the times* T *it took each HPO procedure* $H \in \mathcal{H}$ *executed in strategy* σ *to run. Note that both* $\sigma[\mathcal{L}]$ *and* $\tau_\sigma(\mathcal{L})$ *are random variables, as a function of the randomness of selecting* G *and the actions sampled from* $\sigma(\mathcal{L})$*. For any formula* p *and any* $t \in \mathbb{R}_{>0}$*, we say* $\mathcal{L} \models \Diamond_t p$*, i.e. "*$\mathcal{L}$ *models that it is possible* p *in time* t*," if*

$$\text{there exists a strategy } \sigma \in \Sigma, \text{ such that } \quad \mathbb{P}(\sigma[\mathcal{L}] \models p) = 1 \text{ and } \mathbb{E}[\tau_\sigma(\mathcal{L})] \leq t.$$

Definition 6. *For any formula* p*, we say* $\mathcal{L} \models \mathcal{B}p$*, "*$\mathcal{L}$ *models our belief in* p*", if* $p \in \mathcal{F}(\mathcal{L})$*.*

Definition 7. *Suppose that we are given a naive EHPO procedure* $(\{H\}, \mathcal{F}_n)$*, in which* H *is random search and is the only HPO in our EHPO, and* $\mathcal{F}_n$ *is a "naive" belief function associated with a naive reasoner* $\mathcal{B}_n$*. For any* $K, R \in \mathbb{N}$*, we define the "*(K, R)*-defended" belief function* $\mathcal{F}_*$ *for a skeptical reasoner* $\mathcal{B}_*$ *as the following conclusion-drawing procedure. First,* $\mathcal{F}_*$ *only makes conclusion set* $\mathcal{P}_*$ *from a single log* $\hat{\ell}$ *with* $K * R$ *trials; otherwise, it concludes nothing, outputting* $\emptyset$*. Second,* $\mathcal{F}_*$ *splits the single* $\hat{\ell}$ *into* R *logs* $\ell_1, \ell_2, \ldots, \ell_R$*, each containing* K *independent-random-search trials.*[2] *Finally,* $\mathcal{F}_*$ *outputs the intersection of what the naive reasoner would have output on each log* ℓ_i*,*

$$\mathcal{F}_*(\{\hat{\ell}\}) = \mathcal{P}_* \equiv \mathcal{F}_n(\{\ell_1\}) \cap \mathcal{F}_n(\{\ell_2\}) \cap \cdots \cap \mathcal{F}_n(\{\ell_R\}).$$

Equivalently, $\{\hat{\ell}\} \models \mathcal{B}_* p$ *only if* $\{\ell_i\} \models \mathcal{B}_n p$ *for all* i*.*

[2]This is not generally allowable. $\mathcal{F}_*$ can do this because random-search logs contain interchangeable trials.

B.2 Section 2.2 Appendix: Notes on the Preliminaries

The code for running these experiments can be found at `https://github.com/pasta41/deception`.

B.2.1 Empirical Deception Illustration using Wilson et al.[623]

Why we chose Wilson et al. [623]

We elaborate on why we specifically chose Wilson et al. [623] as our running example of hyperparameter deception. There are four main reasons why we thought this was the right example to focus on for an illustration:

First, the experiment involves optimizers known across ML (e.g. SGD, Adam), a model frequently used for benchmark tasks (VGG16) and a commonly-used benchmark dataset (CIFAR-10). Unlike other examples of hyperparameter deception, one does not need highly-specialized domain knowledge to understand the issue [176, 387]. Second, the paper is exceptionally well-cited and known in the literature, so many folks in the community are familiar with its results. Third, we were certain that we could demonstrate hyperparameter deception before we ran our experiments; we observe that Adam's update rule basically simulates Heavy Ball when its ϵ parameter is set high enough. So, we were confident that we could (at the very least) get Adam to perform as well as Heavy Ball via changing hyper-HPs, which would demonstrate hyperparameter deception. We then found further support for this observation in concurrent work [543], which cited earlier work [122] that also observes this. Fourth, the claim in Wilson et al. [623] is fairly broad. They make a claim about

adaptive vs. non-adaptive optimizers, more generally. If the claim had been narrower – about small ϵ values for numerical stability, then perhaps hyperparameter deception would not have occurred. In general, we note that narrower claims could help avoid deception.

B.2.2 Expanded empirical results

We elaborate on the results we present in Section 2.2.

Experimental setup

We replicate and run a variant of Wilson et al. [623]'s VGG16 experiment on CIFAR-10, using SGD, Heavy Ball, and Adam as the optimizers.

We launch each run on a local machine configured with a 4-core 2.6GHz Inter (R) Xeon(R) CPU, 8GB memory and an NIVIDIA GTX 2080Ti GPU. Following the exact configuration from Wilson et al. [623], we set the mini-batch size to be 128, the momentum constant to be 0.9 and the weight decay to be 0.0005. The learning rate is scheduled to follow a linear rule: The learning rate is decayed by a factor of 10 every 25 epochs. The total number of epochs is set to be 250. For the CIFAR-10 dataset, we apply random horizontal flipping and normalization. Note that Wilson et al. [623] does not apply random cropping on CIFAR-10; thus we omit this step to be consistent with their approach. We adopt the standard cross entropy loss. For each HPO setting, we run 5 times and average the results and include error bars two standard deviations above and below the mean.

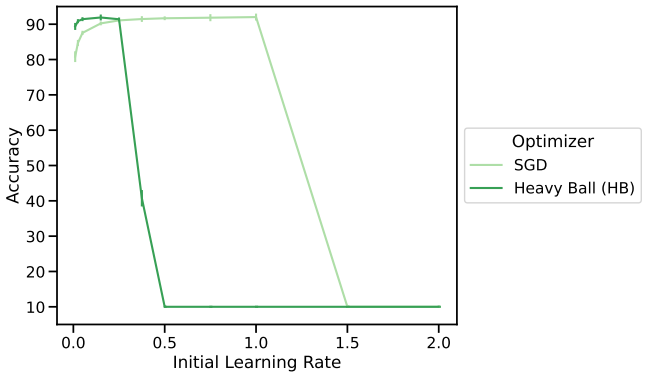


Figure B.1: Full test accuracy results of VGG-16 on CIFAR-10 for SGD and Heavy Ball learning rate (α) HPO. Error bars indicate two standard deviations above and below the mean. Each HPO setting is measured with five replicates. We achieve similar performance as Wilson et al. [623].

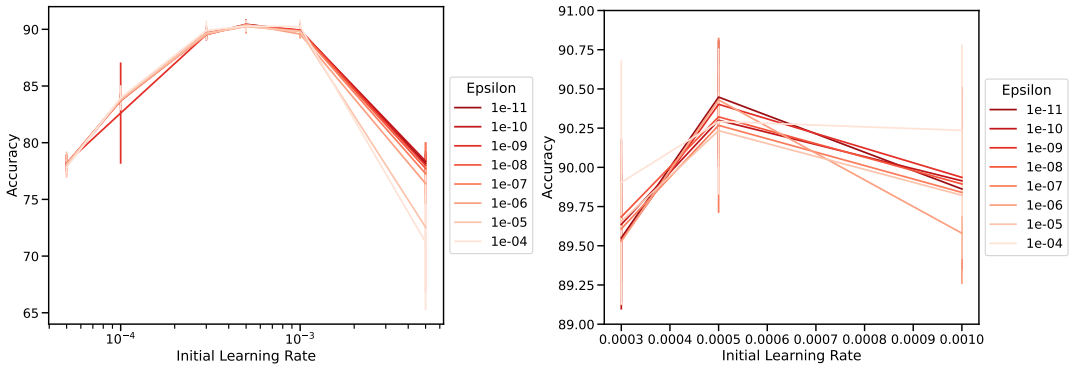


Figure B.2: Tuning over learning rate for different small values of ϵ. On the left, we show a wide range of learning rates tested. On the right, we zoom in on the portion of results where the best test accuracy occurs. These results reflect what Wilson et al. [623] showed, but with tuning over ϵ (small values). Each HP setting is used to train VGG-16 on CIFAR-10 five times, and the error bars represent two standard deviations above and below the mean test accuracy.

Associated results and logs

In line with our notion of a *log* (Definition 1), we provide data tables (Figures B.4, B.5, and B.6) that correspond with our results graphed in the Figures B.1, B.2, B.3.

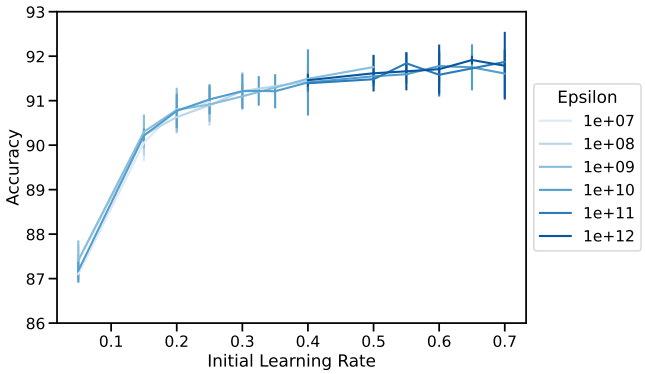


Figure B.3: Results for our expanded search over large ϵ values for Adam. We show test accuracy on CIFAR-10 as a function of different learning rates α for the different large ϵ values. Error bars show two standard deviations above and below mean test accuracy for five replicates for each HP setting.

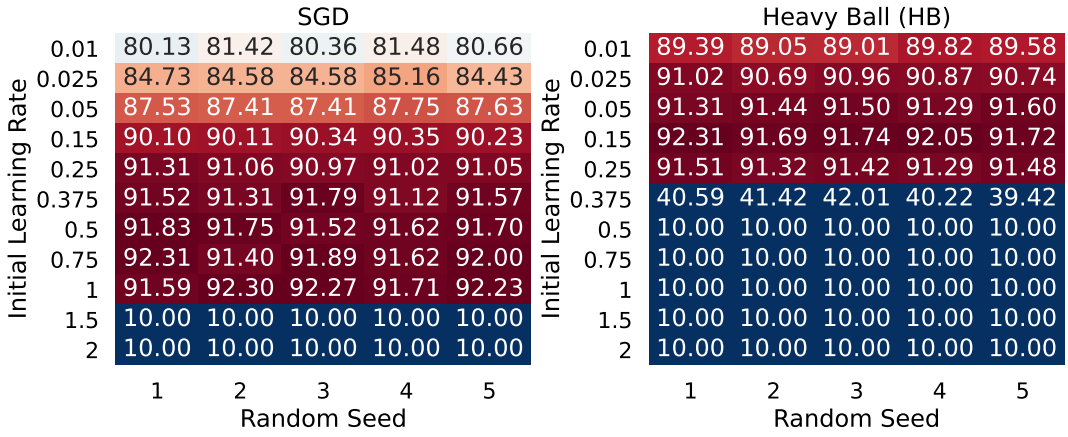


Figure B.4: Heatmap logs of test accuracy of VGG-16 on CIFAR-10 for SGD and Heavy Ball for each initial learning rate and random seed. These logs correspond to the results graphed in Figure B.1.

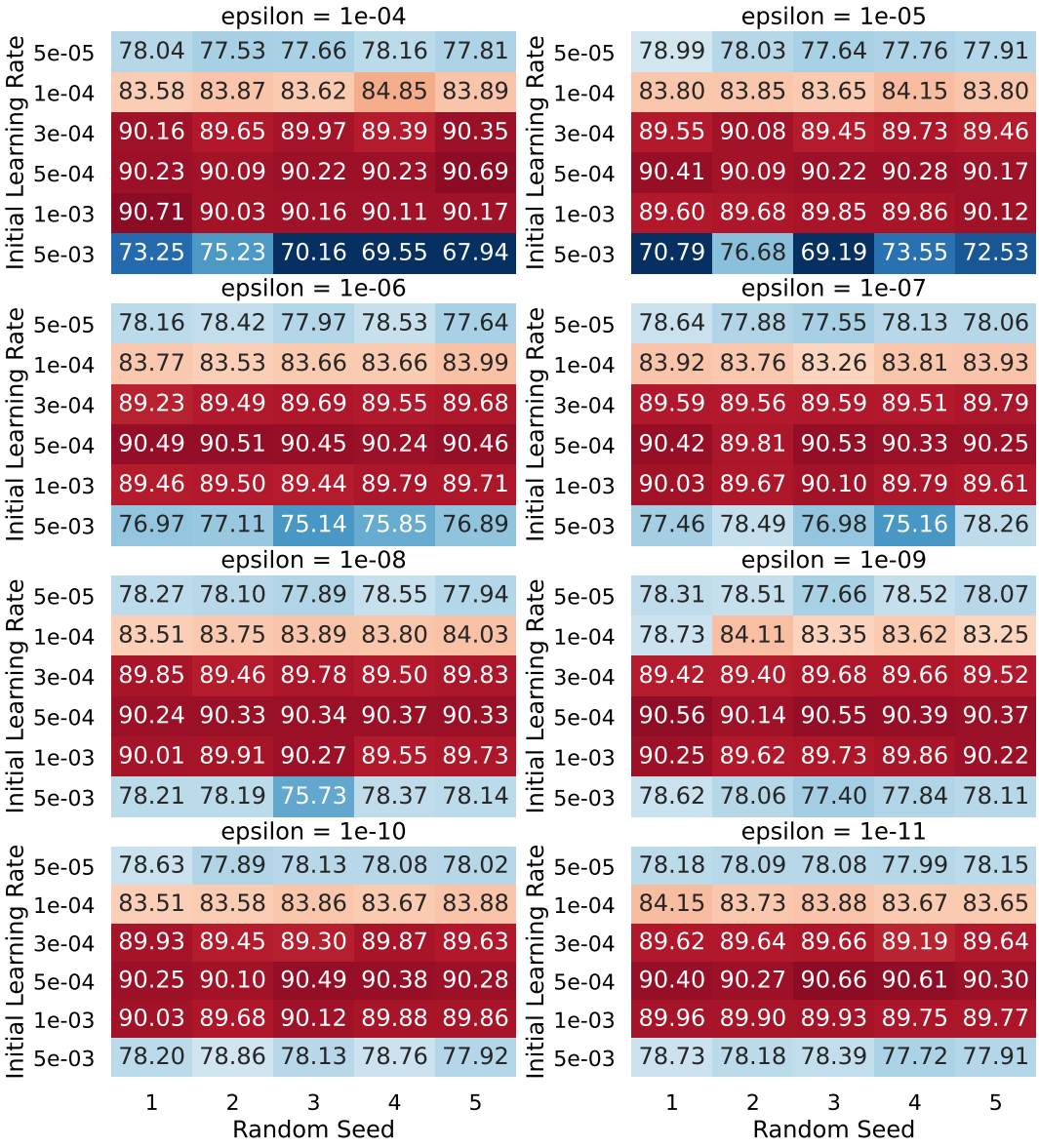


Figure B.5: Heatmap logs of test accuracy of VGG-16 on CIFAR-10 for Adam for each initial learning rate and random seed for different small values of Adam's ϵ HP. These results correspond to those graphed in Figure B.2.

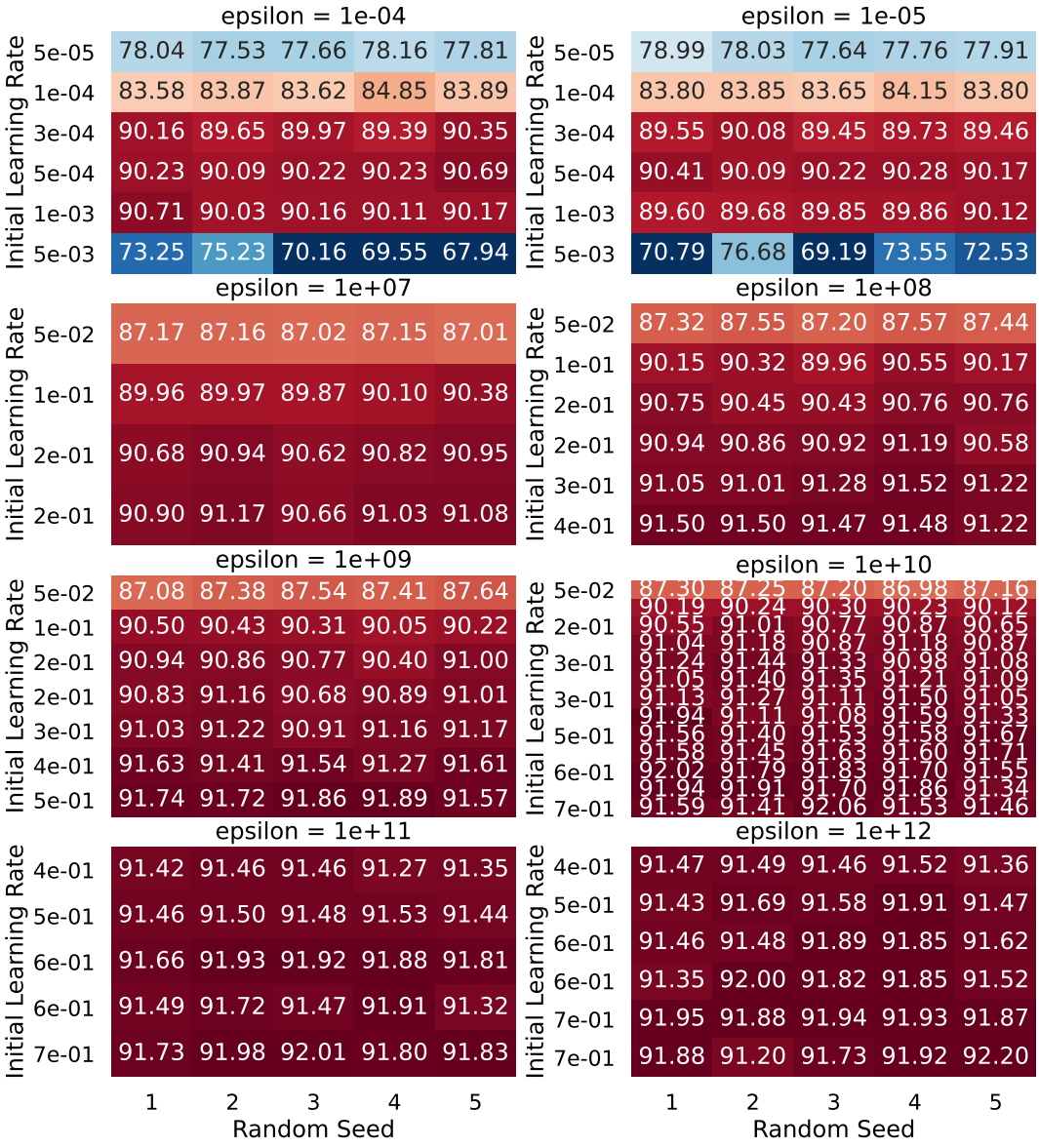


Figure B.6: Heatmap logs of test accuracy of VGG-16 on CIFAR-10 for Adam for each initial learning rate and random seed for different values of Adam's ϵ using our expanded search space. These logs reflect the results graphed in Figure B.3.

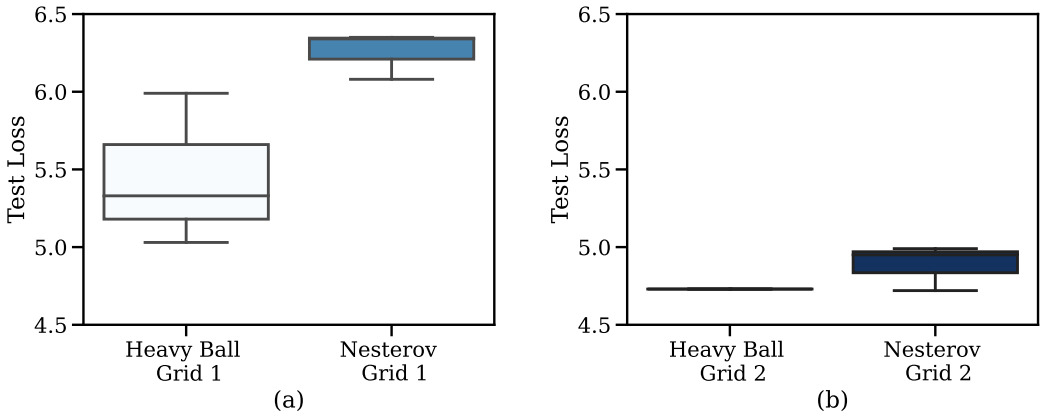


Figure B.7: Demonstrating the possibility of drawing inconsistent conclusions from HPO (what we shorthand *hyperparameter deception*) LSTM on Wikitext-2 using Nesterov and Heavy Ball as the optimizers. Each box plot represents a log. In (a), we use the grid $\alpha = 1, 5, 10, 15, 20, 25, 30, 35, 40$, from which we can reasonably conclude that Nesterov outperforms HB. In (b), we use the grid $\alpha = 10, 20, 30, 40$, from which we can reasonably conclude that HB outperforms Nesterov.

B.2.3 Empirical Deception Illustration using Merity et al. [408]

In addition to the computer vision experiments of Wilson et al. [623], we also show a separate line of experiments from NLP: training an LSTM on Wikitext-2 using Nesterov and Heavy Ball as the optimizers. We illustrate deception (i.e., the possibility of drawing inconsistent conclusions) using two different sets of hyper-HPs to configure HPO grids for tuning the learning rate. We run ten replicates for each optimizer / grid combination (a total of 40 runs). We run these experiments using the same hardware as described in Appendix B.2.2.

B.3 Section 2.3 Appendix: Epistemic Hyperparameter Optimization

B.3.1 Additional concrete interpretations of EHPO

For concision, in the main text we focus on examples of EHPO procedures that compare the performance of different optimizers. However, it is worth noting that our definition of EHPO (Definition 3) is more expansive than this setting. For example, it is possible to run EHPO to compare different models (perhaps, though not necessarily, keeping the optimizer fixed), to draw conclusions about the relative performance of different models on different learning tasks.

B.3.2 Descartes' Evil Demon Thought Experiment

Our formalization was inspired by Descartes' evil genius/demon thought experiment. This experiment more generally relates to his use of systematic doubt in *The Meditations* more broadly. It is this doubt/skepticism (and its relationship to possibility) that we find useful for the framing of an imaginary, worst-case adversary. In particular, we draw on the following quote, from which we came up with the term *hyperparameter deception*:

> *I will suppose...an evil genius, supremely powerful and clever, who has directed his entire effort at deceiving me. I will regard the heavens, the air, the earth, colors, shapes, sounds, and all external things as nothing but the bedeviling hoaxes of my dreams, with which he lays snares for my*

credulity...even if it is not within my power to know anything true, it certainly is within my power to take care resolutely to withhold my assent to what is false, lest this deceiver, however powerful, however clever he may be, have any effect on me. – Descartes

Figure B.8: How the authors imagine the EHPO-running demon

For more on the long (and rich) history of the use of imaginary demons and devils as adversaries — notably a different conception of an adversary than the potential real threats posed in computer security research — we refer the reader to Canales [100].

B.4 Section 2.4 Appendix: Modal Logic Formalization

B.4.1 Further Background on Modal Logic

We first provide the necessary background on modal logic, which will inform the proofs in this appendix (Appendix B.4.1). We then describe our possibility logic—a logic for representing the possible results of the evil demon running

EHPO—and prove that it is a valid modal logic (Appendix B.4.2). We then present a primer on modal belief logic (Appendix B.4.2), and suggest a proof for the validity of combining our modal possibility logic with modal belief logic (Appendix B.4.2).

Axioms from Kripke Semantics

Kripke semantics in modal logic inherits all of the the axioms from propositional logic, which assigns values T and F to each atom p, and adds two operators, one for representing *necessity* ($\Box$) and one for *possibility* ($\Diamond$).

- $\Box p$ reads "It is necessary that p".
- $\Diamond p$ reads "It is possible that p".

The $\Diamond$ operator is just syntactic sugar, as it can be represented in terms of $\neg$ and $\Box$:

$$\Diamond p \equiv \neg\Box\neg p \tag{B.1}$$

which can be read as:

"It is possible that p" is equivalent to "It is not necessary that not p."

The complete set of rules is as follows:

- Every atom p is a sentence.
- If D is a sentence, then

  - $\neg D$ is a sentence.
  - $\Box D$ is a sentence.
  - $\Diamond D$ is a sentence.
- If D and E are sentences, then
  - $D \wedge E$ is a sentence.
  - $D \vee E$ is a sentence.
  - $D \rightarrow E$ is a sentence.
  - $D \leftrightarrow E$ is a sentence
- $\Box(\mathcal{D} \rightarrow \mathcal{E}) \rightarrow (\Box\mathcal{D} \rightarrow \Box\mathcal{E})$ (Distribution)
- $\mathcal{D} \rightarrow \Box\mathcal{D}$ (Necessitation)

Possible Worlds Semantics

Modal logic introduces a notion of *possible worlds*. Broadly speaking, a possible world represents the state of how the world *is* or potentially *could be* [118, 225]. Informally, $\Box D$ means that D is true at *every* world (Equation B.2); $\Diamond D$ means that D is true at *some* world (Equation B.3).

Possible worlds give a different semantics from more familiar propositional logic. In the latter, we assign truth values $\{T, F\}$ to propositional variables $p \in \mathcal{P}$, from which we can construct and evaluate sentences $D \in \mathcal{D}$ in a truth table. In the former, we introduce a set of possible worlds, $\mathcal{W}$, for which each $w \in \mathcal{W}$ has own truth value for each p. This means that the value of each p can differ across different worlds w. Modal logic introduces the idea of valuation function,

$$\mathcal{V} : (\mathcal{W} \times \mathcal{D}) \rightarrow \{T, F\}$$

to assign truth values to logical sentences at different worlds. This in turn allows us to express the formulas, axioms, and inference rules of propositional logic in terms of $\mathcal{V}$. For example,

$$\mathcal{V}(w, \neg D) = T \leftrightarrow \mathcal{V}(w, D) = F$$

There are other rules that each correspond to a traditional truth-table sentence evaluation, but conditioned on the world in which the evaluation occurs. We omit these for brevity and refer the reader to Challas [118].

We do include the valuation rules for the $\Box$ and $\Diamond$ operators that modal logic introduces (Equations B.2 & B.3). To do so, we need to introduce one more concept: The accessibility relation, $\mathcal{R}$. $\mathcal{R}$ provides a frame of reference for one particular possible world to access other possible worlds; it is a way from moving from world to world. So, for an informal example, $\mathcal{R}w_1w_2$ means that w_2 is possible relative to w_1, i.e. we can reach w_2 from w_1. Such a relation allows for a world to be possible relative potentially to some worlds but not others. More formally,

$$R \subseteq \mathcal{W} \times \mathcal{W}$$

Overall, the important point is that we have a collection of worlds $\mathcal{W}$, an accessibility relation $\mathcal{R}$, and a valuation function $\mathcal{V}$, which together defines a Kripke model, which captures this system:

$$\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle$$

Finally, we can give the valuation function rules for $\Box$ and $\Diamond$:

$$\mathcal{V}(w, \Box D) = T \leftrightarrow \forall w', (\mathcal{R}ww' \rightarrow \mathcal{V}(w', D) = T) \tag{B.2}$$

$$\mathcal{V}(w, \Diamond D) = T \leftrightarrow \exists w', (\mathcal{R}ww' \wedge \mathcal{V}(w', D) = T) \tag{B.3}$$

Informally, for $\Box D$ to be true in a world, it must be true in every possible world that is reachable by that world. For $\Diamond D$ to be true in a world, it must be true in some possible world that is reachable by that world.

B.4.2 Our Multimodal Logic Formulation

A Logic for Reasoning about the Conclusion of EHPO

As in Section 2.4, we can define the well-formed formulas of our indexed modal logic[3] recursively in Backus-Naur form, where t is any real number and P is any atomic proposition

$$\kappa \coloneqq P \mid \neg\kappa \mid \kappa \wedge \kappa \mid \Diamond_t \kappa \tag{B.4}$$

where κ is a well-formed formula.

As we note in Section 2.4, where we first present this form of defining modal-logic, $\Box$ is syntactic sugar, with $\Box p \equiv \neg\Diamond\neg p$ (which remains true for our indexed modal logic). Similarly, "or" has $p \vee q \equiv \neg(\neg p \wedge \neg q)$ and "implies" has $p \rightarrow q \equiv \neg p \vee q$, which is why we do not include them for brevity in this recursive definition.

We explicitly define the relevant semantics for $\Diamond_t$ for reasoning about the demon's behavior in running EHPO. For clarity, we replicate that definition of the

[3]For an example of another indexed modal logic concerning probability, please refer to Heifetz and Mongin [267].

semantics of expressing the possible outcomes of EHPO conducted in bounded time (Definitions 4 & 5, respectively) below:

Definition. *A randomized* ***strategy*** $\boldsymbol{\sigma}$ *is a function that specifies which action the demon will take. Given* $\mathcal{L}$*, its current set of logs,* $\boldsymbol{\sigma(\mathcal{L})}$ *gives a distribution over concrete actions, where each action is either 1) running a new* H *with its choice of hyper-HPs* c *and seed* r *2) erasing some logs, or 3) returning. We let* Σ *denote the set of all such strategies.*

We can now define what the demon can reliably bring about, in terms of executing a strategy in bounded time:

Definition. *Let* $\boldsymbol{\sigma[\mathcal{L}]}$ *denote the logs output from executing strategy* σ *on logs* $\mathcal{L}$*, and let* $\boldsymbol{\tau_\sigma(\mathcal{L})}$ *denote the total time spent during execution.* $\tau_\sigma(\mathcal{L})$ *is equivalent to the sum of the times* T *it took each HPO procedure* $H \in \mathcal{H}$ *executed in strategy* σ *to run. Note that both* $\sigma[\mathcal{L}]$ *and* $\tau_\sigma(\mathcal{L})$ *are random variables, as a function of the randomness of selecting* G *and the actions sampled from* $\sigma(\mathcal{L})$*. For any formula* p *and any* $t \in \mathbb{R}_{>0}$*, we say* $\mathcal{L} \models \Diamond_t p$*, i.e. "*$\mathcal{L}$ *models that it is possible* p *in time* t*," if*

$$\text{there exists a strategy } \sigma \in \Sigma, \text{ such that } \quad \mathbb{P}(\sigma[\mathcal{L}] \models p) = 1 \text{ and } \mathbb{E}[\tau_\sigma(\mathcal{L})] \leq t.$$

A Possible Worlds Interpretation

Drawing on the possible worlds semantics that modal logic provides (Section B.4.1), we can define specific possible worlds semantics for our logic for expressing the actions of the demon in EHPO from above.

Definition 8. *A* ***possible world*** *represents the set of logs* $\mathcal{L}$ *the demon has produced at time* $\tau_\sigma(\mathcal{L})$ *(i.e., after concluding running EHPO) and the set of formulas* $\mathcal{P}$ *that are modeled from* $\mathcal{L}$ *via* $\mathcal{F}$*.*

Therefore, different possible worlds represent the states that *could have existed* if the evil demon had executed different strategies (Definition 4). In other words, if it had performed EHPO with different learning algorithms, different HPO procedures, different hyper-hyperparameter settings, different amounts of time (less than the total upper bound), different learning tasks, different models, etc... to produce a different set of logs $\mathcal{L}$ and corresponding set of conclusions $\mathcal{P}$.

In this formulation, the demon has knowledge of all possible worlds; it is trying to fool us about the relative performance of algorithms by showing as an intentionally deceptive world. Informally, moving from world to world (via an accessibility relation) corresponds to the demon running more passes of HPO to produce more logs to include in $\mathcal{L}$.

Syntax and Semantics for the Logic Modeling the Demon Running the EHPO

We provide proofs and intuitions of the axioms of our EHPO logic in this section, based on a correspondence with un-indexed modal logic.

We remind the reader that the following are the axioms of our indexed modal logic:

$$\vdash (p \to q) \to (\Diamond_t p \to \Diamond_t q) \qquad \textit{(necessitation + distribution)}$$

$$p \to \Diamond_t p \qquad \textit{(reflexivity)}$$

$$\Diamond_t \Diamond_s p \to \Diamond_{t+s} p \qquad \textit{(transitivity)}$$

$$\Diamond_s \Box_t p \to \Box_t p \qquad \textit{(symmetry)},$$

$$\Diamond_t (p \wedge q) \to (\Diamond_t p \wedge \Diamond_t q) \qquad \textit{(distribution over } \wedge \textit{)}$$

In short, to summarize these semantics—the demon has knowledge of all possible hyper-hyperparameters, and it can pick whichever ones it wants to run EHPO within a bounded time budget t to realize the outcomes it wants: $\Diamond_t p$ means it can realize p.

We inherit distribution and necessitation from un-indexed modal logic; they are axiomatic based on Kripke semantics. We provide greater intuition and proofs below.

Notes on necessitation for $\Box_t$:

Necessitation for our indexed necessary operator can be written as follows:

$$\vdash p \rightarrow \Box_i p$$

As we note in Section 2.4, $\vdash$ just means here that p is a theorem of propositional logic. So, if p is a theorem, then so is $\Box_t p$. By theorem we just mean that p is provable by our axioms (these being the only assumptions we can use); so whenever p fits this definition, we can say $\Box_t p$.

For our semantics, this just means that when p is a theorem, it is necessary that p in time t.

Distribution for $\Box_t$:

$$\Box_t(p \rightarrow q) \rightarrow (\Box_t p \rightarrow \Box_t q)$$

We provide three ways to verify distribution over implication for $\Box_t$. From this, we will prove distribution over implication for $\Diamond_t$

A. The first follows from an argument about the semantics of possible worlds

from the Kripke model of our system (Sections B.4.1 & B.4.2).

i. It is fair to reason that distribution is self-evident given the definitions of implication ($\rightarrow$, formed from $\neg$ and $\vee$ in our syntax for well-formed formulas for our EPHO logic, given at (B.4) and necessity ($\Box_t$, formed from $\neg$ and $\Diamond_t$ in our syntax for well-formed formulas for our EHPO logic, given at (B.4)).

ii. Similarly, we can further support this via our semantics of possible worlds.

We can understand $\Box_t p$ to mean, informally, that it an adversary does adopt a strategy σ that is guaranteed to cause the desired conclusion p to be the case while take at most time t in expectation. Formally, as an "necessary" analog to the semantics of $\Diamond_t$ given in Definition 5:

For any formula p, we say $\mathcal{L} \models \Box_t p$ if and only if

$$\forall \sigma \in \Sigma,\ \mathbb{P}(\sigma[\mathcal{L}] \models p) = 1 \ \wedge\ \mathbb{E}[\tau_\sigma(\mathcal{L})] \leq t.$$

Given $p \rightarrow q$ is true in **all accessible worlds** (i.e, the definition of necessary), then we can say that q is true in all accessible worlds whenever p is true in all accessible worlds. As in i. above, this just follows / is axiomatic from the definitions of necessity and implication for Kripke semantics.

B. We can also prove distribution by contradiction.

i. Suppose that the distribution axiom does not hold. That is, the hypothesis

$$\Box_t(p \rightarrow q)$$

is true and the conclusion

$$\Box_t p \rightarrow \Box_t q$$

is false.

ii. By similar reasoning, from above $\Box_t p \rightarrow \Box_t q$ being false, we can say that $\Box_t p$ is true and $\Box_t q$ is false.

iii. We can use Modal Axiom M (reflexivity, proven in the next section) to say $\Box_t p \rightarrow p$. Since $\Box_t p$ is true, we can use *modus ponens* to determine that p is true.

iv. We can also say

$$\Box_t(p \rightarrow q) \rightarrow (p \rightarrow q) \qquad (\textit{By Modal Axiom M (reflexivity)})$$

v. Since we $\Box_t(p \rightarrow q)$ is true from above, we can conclude via *modus ponens* that $p \rightarrow q$ must also be true.

vi. We concluded above that p is true, so we can again use *modus ponens* and the fact that $p \rightarrow q$ is true to conclude that q is true.

vii. By necessitation, we can then also say $q \rightarrow \Box_t q$, and conclude that $\Box_t q$ is true. This is a contradiction, as above we said that $\Box_t$ is false.

viii. Therefore, by contradiction, $\Box_t(p \rightarrow q) \rightarrow (\Box_t p \rightarrow \Box_t q)$ is proved.

C. We can separately take an intuitionistic approach to verify the distribution axiom [30, 57]:

i. Let b be an **actual proof** of $p \rightarrow q$ so that we can say $a.b$ is a proof of $\Box_t(p \rightarrow q)$.

ii. Let d be an **actual proof** of p so that we can say $c.d$ is a proof of $\Box_t p$.

iii. From i. and ii., $b(d)$ is an **actual proof** of q, i.e. b (an actual proof of $p \rightarrow q$) is supplied d (an actual proof of p), and therefore can conclude q via an actual proof.

iv. From iii., we can say this results in a proof of $\Box_t q$, i.e. $e.[b(d)]$.

v. The above i.-iv. describes a function, $f : a.b \rightarrow f_{(a.b)}$. In other words, given **any proof** $a.b$ (i.e., of $\Box_t(p \rightarrow q)$) we can return function $f_{(a.b)}$, which turns **any proof** $c.d$ (i.e., of $\Box_t p$) into a proof $e.[b(d)]$ (i.e., of $\Box_t q$).

vi. $f_{(a.b)}$ is thus a proof of $\Box_t p \rightarrow \Box_t q$.

vii. From i.-vi., we gone from $a.b$ (a proof of $\Box_t(p \rightarrow q)$) to a proof of $\Box_t p \rightarrow \Box_t q$, i.e. have intuitionistically shown that $\Box_t(p \rightarrow q) \rightarrow (\Box_t p \rightarrow \Box_t q)$

Distribution and $\Diamond_t$: We provide the following axiom in our logic:

$$\vdash (p \rightarrow q) \rightarrow (\Diamond_t p \rightarrow \Diamond_t q) \qquad (\textit{necessitation and distribution})$$

and we now demonstrate it to be valid.

$$\begin{aligned}
\vdash (p \rightarrow q) &\rightarrow \Box_t(p \rightarrow q) && (\textit{necessitation}) \\
&\rightarrow \Box_t(\neg q \rightarrow \neg p) && (\textit{modus tollens}) \\
&\rightarrow (\Box_t \neg q \rightarrow \Box_t \neg p) && (\textit{distribution}) \\
&\rightarrow (\neg \Box_t \neg p \rightarrow \neg \Box_t \neg q) && (\textit{modus tollens}) \\
&\rightarrow (\Diamond_t p \rightarrow \Diamond_t q) && (\Diamond_t a \equiv \neg \Box_t \neg a)
\end{aligned}$$

This concludes our proof, for how the axioms are jointly stated.

Further, we could also say

$$(p \rightarrow q) \rightarrow \Diamond_t(p \rightarrow q) \qquad \textit{(Modal axiom M (reflexivity))}$$

And therefore also derive distribution over implication for possibility:

$$\Diamond_t(p \rightarrow q) \rightarrow (\Diamond_t p \rightarrow \Diamond_t q)$$

Modal Axiom M: Reflexivity

$$p \rightarrow \Diamond_t p$$

This axiom follows from how we have defined the semantics of our indexed modal logic (Definition 5). It follows from the fact that the demon could choose to do nothing.

We can provide a bit more color to the above as follows:

We can also derive this rule from necessitation, defined above (and from the general intuition / semantics of modal logic that necessity implies possibility). First, we can say that necessity implies possibility. We can see this a) from a possible worlds perspective and b) directly from our axioms. From a possible worlds perspective, this follows from the definition of the operators. Necessity means that there is truth at every accessible possible world, while possibility means there is truth at some accessible possible world, which puts that possible truth in time t as a subset of necessary truth in time t. From the axioms, we

verify

$\Box_t p \rightarrow \Diamond_t p$ (*Theorem to verify, which also corresponds to Modal Axiom D (serial)*)

$\neg\Box_t p \vee \Diamond_t p$ (*Applying* $p \rightarrow q$ *is equiv.* $\neg p \vee q$)

$\Diamond_t \neg p \vee \Diamond_t p$ (*By modal conversion,* $\neg\Box_t p \rightarrow \Diamond_t \neg p$)

(*Which for our semantics is tautological*)

That is, in time t it is possible that p or it is possible that p, which allows for us also to not conclude anything (in the case that the demon chooses to do nothing).

We can then say,

$(\Box_t p \rightarrow p) \rightarrow \Diamond_t p$ (*By necessitation and* $\Box_t p \rightarrow \Diamond_t p$)

$p \rightarrow \Diamond_t p$ (*By concluding p from necessitation*)

Another way to understand this axiom is again in terms of possible worlds. We can say in our system that every world is possible in relation to itself. This corresponds to the accessibility relation $\mathcal{R}ww$. As such, an equivalent way to model reflexivity is in terms of the following:

$$\Box_t p \rightarrow p$$

That is, if $\Box_t p$ holds in world w, then p also holds in world w, as is the case for $\mathcal{R}ww$. We can see this by proving $\Box_t p \rightarrow p$ by contradiction. Assuming this were false, we would need to construct a world w in which $\Box_t p$ is true and p is false. If $\Box_t p$ is true at world w, then by definition p is true at every world that w accesses. For our purposes, this holds, as $\Box_t p$ means that it is necessary for

p to be the case in time t; any world that we access from this world w (i.e. by say increasing time, running more HPO) would require p to hold. Since $\mathcal{R}ww$ means that w accesses itself, that means that p must also be true at w, yielding the contradiction.

Modal Axiom 4: Transitivity

$$\Diamond_t \Diamond_s p \rightarrow \Diamond_{t+s} p \tag{B.5}$$

We can similarly understand transitivity to be valid intuitively from the behavior of the demon and in relation to the semantics of our possible worlds. We do an abbreviated treatment (in relation to what we say for reflexivity above) for brevity.

In terms of the demon, we note that in our semantics $\Diamond_t p$ means that it is possible for the demon to bring about conclusion p via its choices in time t. Similarly, we could say the same for $\Diamond_s p$; this means it is possible for the demon to bring about conclusion p in time s. If it is possible in time t that it is possible in time s to bring about p, this is equivalent in our semantics to saying that it is possible in time $t + s$ to bring about conclusion p.

We can understand this rule (perhaps more clearly) in terms of possible worlds and accessibility relations.

For worlds w_n,

$$\forall w_1, \forall w_2, \forall w_3, \mathcal{R}w_1w_2 \wedge \mathcal{R}w_2w_3 \rightarrow \mathcal{R}w_1w_3$$

In other words, this accessibility relation indicates that if w_1 accesses w_2 and if w_2 accesses w_3, then w_1 accesses w_3.

For understanding this in terms of relative possibility, we could frame this as, if w_3 is possible relative to w_2 and if w_2 is possible relative to w_1, then w_3 is possible relative to w_1. For our semantics of the demon, this means that in some time if in some time b we can get to some possible world w_3 from when we're in w_2 and in time a we can get to some possible world w_2 when we're in w_1, then in time $a + b$ we can get to w_3 from w_1

This axiom is akin to us regarding a string of exclusively possible or exclusively necessary modal operators as just one possible or necessary modal operator, respectively; we regard then regard sum of times as the amount of time it takes to bring about p (again, being necessary or possible, respectively).

Modal Axiom 5: Symmetry

$$\Diamond_s \Box_t p \rightarrow \Box_t p \tag{B.6}$$

We can similarly understand that our modal logic is symmetric; this is valid intuitively from the behavior of the demon. We further abbreviate our treatment for brevity. In terms of the demon, we note that in our semantics $\Diamond_s p$ means that it is possible for the demon to bring about conclusion p via its choices in time s. We can also say $\Box_t p$ means that it is necessary for the demon to bring about p in time t. If it is possible in time s that it is necessary in time s to bring about p, this is equivalent in our semantics to saying that it is necessary in time t to bring about conclusion p. In other words, we can disregard would could have possibly happened in time s from the demon's behavior and only regard what was necessary in time t for the demon to do in order to bring about p.

As another example, consider our reduction of $\Diamond_t \neg \Diamond_t p$ to $\neg \Diamond_t p$ in our proof for deriving a defended reasoner in Section 2.5. While the intuitive English

reading ("It's possible that it's not possible that p") does not seem equivalent to this reduction ("It's not possible that p), it is in fact valid for our semantics. Think of this in terms of the demon. If p cannot be brought about in time t in expectation (where t is a reasonable upper bound on compute time), then that's the end of it; it doesn't matter which operators come before it (any number of $\Diamond_t$ or $\Box_t$). Adding possibility or necessity before that condition doesn't change that fact that it, for that upper bound t, it is not possible to bring about p.

This axiom is akin to us just regarding the rightmost modal operator when we have a mix of modal operators applied iteratively; we can disregard what was possible or necessary in the time prior to the rightmost operator, and say that what we can say about a sentence p (whether it is possible or necessary) just relates to how much time the last operator required to bring about p.

Derived axioms

We can similarly derive other axioms of our indexed modal logic, form the axioms above. Notably,

$\Box_t$ distributes over $\wedge$

$\Box_t(p \wedge q) \rightarrow (\Box_t p \wedge \Box_t q)$	($\Box_t$ *distributes over* $\wedge$)
Inner proof 1	
$p \wedge q$	
p	
$(p \wedge q) \rightarrow p$	
$\Box_t((p \wedge q) \rightarrow p)$	(*Necessitation*)
$\Box_t(p \wedge q) \rightarrow \Box_t p$	(*Distribution*)
$\Box_t p$	(*By assuming the hypothesis*)
Inner proof 2	
$p \wedge q$	
q	
$(p \wedge q) \rightarrow q$	
$\Box_t((p \wedge q) \rightarrow q)$	(*Necessitation*)
$\Box_t(p \wedge q) \rightarrow \Box_t q$	(*Distribution*)
$\Box_t q$	(*By assuming the hypothesis*)
$\Box_t p \wedge \Box_t q$	(*By inner proof 1, inner proof 2, assuming the hypothesis*)

We can show a similar result for $\Diamond_t$ and $\wedge$, omitted for brevity.

$\Diamond_t$ distributes over $\vee$

$$\Diamond_t(p \vee q) \rightarrow (\Diamond_t p \vee \Diamond_t q) \qquad (\Diamond \textit{ distributes over } \vee)$$

$$\neg\Box_t\neg(p \vee q) \rightarrow (\Diamond_t p \vee \Diamond_t q) \qquad (\Diamond_t a \equiv \neg\Box_t\neg a)$$

$$\neg\Box_t(\neg p \wedge \neg q) \rightarrow (\Diamond_t p \vee \Diamond_t q) \qquad (\neg(a \vee b) \equiv (\neg a \wedge \neg b))$$

$$\neg(\Box_t\neg p \wedge \Box_t\neg q) \rightarrow (\Diamond_t p \vee \Diamond_t q) \qquad (\Box_t \textit{ distributes over } \wedge)$$

$$(\neg\Box_t\neg p \vee \neg\Box_t\neg q) \rightarrow (\Diamond_t p \vee \Diamond_t q) \qquad (\neg(a \wedge b) \equiv (\neg a \vee \neg b))$$

$$(\Diamond_t p \vee \Diamond_t q) \rightarrow (\Diamond_t p \vee \Diamond_t q) \qquad (\Diamond_t a \equiv \neg\Box_t\neg a)$$

We can show a similar result for $\Box_t$ and $\vee$, omitted for brevity.

Syntax and Semantics for the Logic of our Belief in EHPO Conclusions

The logic of belief is a type of modal logic, called doxastic logic [277], where the modal operator $\mathcal{B}$ is used to express belief[4] Different types of reasoners can be defined using axioms that involve $\mathcal{B}$ [550].

We can formulate the doxastic logic of belief in Backus-Naur form:

For any atomic proposition P, we define recursively a well-formed formula ϕ as

$$\phi := P \mid \neg\phi \mid \phi \wedge \phi \mid \mathcal{B}\phi \tag{B.7}$$

where $\mathcal{B}$ means "It is believed that ϕ". We interpret this recursively where p is the base case, meaning that ϕ is p if it is an atom, $\neg\phi$ is well-formed if ϕ is

[4]Computer scientists do not tend to distinguish between the logic of knowledge (epistemic) and the logic of belief (doxastic) [527].

well-formed. We can also define $\vee$, $\rightarrow$, and $\leftrightarrow$ from $\neg$ and $\wedge$, as in propositional logic.

As stated in the belief logic portion of Section 2.4, we model a consistent Type 1 reasoner [550], which has access to all of propositional logic, has their beliefs logical closed under *modus ponens*, and does not derive contradictions. In other words, we have the following axioms:

$$\neg(\mathcal{B}p \wedge \mathcal{B}\neg p) \equiv \mathcal{B}p \rightarrow \neg\mathcal{B}\neg p$$

which is the consistency axiom,

$$\vdash p \rightarrow \mathcal{B}p$$

which is akin to Necessitation above in Section B.4.1 and means that we believe all tautologies, and

$$\mathcal{B}(p \rightarrow q) \rightarrow (\mathcal{B}p \rightarrow \mathcal{B}q)$$

which means that belief distributes over implication. This notably does not include

$$\mathcal{B}p \rightarrow p$$

which essentially means that we do not allow for believing p to entail concluding p. This corresponds to us actually wanting to run hyperparameter optimization before we conclude anything to be true. We do not just want to conclude something to be true based only on *a priori* information. This is akin to picking folkore parameters and concluding they are optimal without running hyperparameter optimization.

Combining Logics

It is a well known result that we can combine modal logics to make a multi-modal logic [525]. In particular, we refer the reader to results on *fusion* [578].

For a brief intuition, we are able to combine our EHPO logic with belief logic since we are operating over the same set of possible worlds. The results of running EHPO produce a particular possible world, to which we apply our logic of belief in order to reason about the conclusions drawn in that world.

Our Combined, Multimodal Logic and Expressing Hyperparameter Deception

We develop the following multimodal logic, which we also state in Section 2.4:

$$\psi := P \mid \neg\psi \mid \psi \wedge \psi \mid \Diamond_t \psi \mid \mathcal{B}\psi$$

Axioms

We give this multimodal logic semantics to express our t-non-deceptiveness axiom, which we repeat below for completeness:

For any formula p,

$$\neg\left(\Diamond_t \mathcal{B}p \wedge \Diamond_t \mathcal{B}\neg p\right)$$

We can similarly express the t-non-deceptiveness axiom:

For any formula p,

$$\Diamond_t \mathcal{B}p \rightarrow \neg\Diamond_t \mathcal{B}\neg p$$

We can also express a *t*-deceptiveness-axiom:

For any formula p,

$$\Diamond_t \mathcal{B}p \wedge \Diamond_t \mathcal{B}\neg p$$

To reiterate, *multimodal* just means that we have multiple different modes of reasoning, in this case our $\Diamond_t$ semantics for the demon doing EHPO and our consistent Type 1 reasoner operator $\mathcal{B}$.

Given a reasonable maximum time budget *t*, we say that EHPO is *t*-non-deceptive if it satisfies all of axioms above. Moreover, based on this notion of *t*-non-deceptiveness, we can express what it means to have a defense to being deceived.

Some notes on strength of belief and belief update

There are potentially interesting connections between our work on defending against hyperparameter deception and belief update [204]. Notably, one could view our notion of skeptical belief as related to work done on "strength of belief" and belief update, or dynamic doxastic logic [326, 527, 592]. Instead of picking an EHPO runtime *a priori* and then running a defended EHPO and at the end evaluating whether or not we believe the conclusions we draw, we could iteratively update and test our belief and terminate if a certain belief threshold is met. In such quantitative theories of belief change, the degree of acceptance of a sentence is represented by a numerical value. Those numerical values can be updated in light of new information (so-called "soft" information updates) [36, 591]. Exploring this is out of scope for our work here, but could be an interesting future research direction for how to reason about empirical results that

imply inconsistent conclusions.

B.5 Section 2.5 Appendix: Notes on Defenses

B.5.1 Proving a defended reasoners

Suppose that we have been drawing conclusions using some "naive" belief operator $\mathcal{B}_{\text{n}}$ (based on a conclusion function $\mathcal{F}_{\text{n}}$) that satisfies the axioms of Section 2.4.3. We want to use $\mathcal{B}_{\text{n}}$ to construct a new operator $\mathcal{B}_*$, which is guaranteed to be deception-free. One straightforward way to do this is to define the belief operator $\mathcal{B}_*$ such that for any statement p,

$$\mathcal{B}_* p \;\equiv\; \mathcal{B}_{\text{n}} p \;\wedge\; \neg \Diamond_t \mathcal{B}_{\text{n}} \neg p.$$

That is, we conclude p only if both our naive reasoner would have concluded p, and it is impossible for an adversary to get it to conclude $\neg p$ in time t. This enables us to show t-non-deceptiveness, following directly from the axioms in a proof by contradiction: Suppose $\mathcal{B}_*$ can be deceived, i.e. $\Diamond_t \mathcal{B}_* p \wedge \Diamond_t \mathcal{B}_* \neg p$ is True:

			Rule
$\Diamond_t \mathcal{B}_* p$	$\equiv$	$\Diamond_t (\mathcal{B}_{\text{n}} p \wedge \neg \Diamond_t \mathcal{B}_{\text{n}} \neg p)$	Applying $\Diamond_t$ to the definition of $\mathcal{B}_* p$ (2.1)
	$\rightarrow$	$\Diamond_t (\neg \Diamond_t \mathcal{B}_{\text{n}} \neg p)$	Reducing a conjunction to either of its terms: $(a \wedge b) \rightarrow b$
	$\rightarrow$	$\neg \Diamond_t \mathcal{B}_{\text{n}} \neg p$	Symmetry; dropping all but the right-most operator: $\Diamond_t (\Diamond_t a) \rightarrow \Diamond_t a$

We provide more detail on these transformations than we do in the main text. The first application is simple; we just put parentheses around our definition of $\mathcal{B}_*$, and apply $\Diamond_t$ to it. The second step is also simple. We apply a change to

whats inside the parentheses, i.e. just the definition of $\mathcal{B}_*$. Because this is a conjunction, in order for it to be true, both components have to be true. So, we can reduce the conjunction to just it's second term.

The part that is more unfamiliar is the application of Modal Axiom 5 (Symmetry) to reduce the number of $\Diamond_t$ operators. We provide this example above in Section B.4, where we explain why Modal Axiom 5 holds for our EHPO logic semantics. We reiterate here for clarity:

While the intuitive English reading ("It's possible that it's not possible that p") does not seem equivalent to this reduction ("It's not possible that p), it is in fact valid for our semantics. Think of this in terms of the demon. If p cannot be brought about in time t in expectation (where t is a reasonable upper bound on compute time), then that's the end of it; it doesn't matter which operators come before it (any number of $\Diamond_t$ or $\Box_t$). Adding possibility or necessity before that condition doesn't change that fact that it, for that upper bound t, it is not possible to bring about p.

We then pause to apply our axioms to the right side of the conjunction, $\Diamond_t \mathcal{B}_* \neg p$:

			Rule
$\Diamond_t \mathcal{B}_* \neg p$	$\equiv$	$\Diamond_t (\mathcal{B}_{\mathrm{n}} \neg p \wedge \neg \Diamond_t \mathcal{B}_{\mathrm{n}} p)$	Applying $\Diamond_t$ to the definition of $\mathcal{B}_* \neg p$ (1)
	$\rightarrow$	$\Diamond_t \mathcal{B}_{\mathrm{n}} \neg p \wedge \Diamond_t \neg \Diamond_t \mathcal{B}_{\mathrm{n}} p$	Distributing $\Diamond_t$ over $\wedge$: $\Diamond_t(a \wedge b) \rightarrow (\Diamond_t a \wedge \Diamond_t b)$
	$\rightarrow$	$\Diamond_t \mathcal{B}_{\mathrm{n}} \neg p$	Reducing a conjunction to either of its terms: $(a \wedge b) \rightarrow a$

This transformation is much like the one above. We similarly apply $\Diamond_t$ to the definition of $\mathcal{B}_*$.

We then distribute $\Diamond_t$ over the definition, which holds for our logic since possibility distributes over and. We prove this for our logic in Section B.4, and provide an intuitive explanation here. If it is possible in time t to bring about a particular formula, then it must also be possible to bring about the sub-conditions that compose that formula in time t. If this were not the case, then we would not be able to satisfy bringing about the whole formula in time t.

Lastly, as in the first example, we reduce the conjunction to one of its terms (this time taking the first, rather than the second).

We now bring both sides of the conjunction back together: $\Diamond_t \mathcal{B}_* p \wedge \Diamond_t \mathcal{B}_* \neg p \equiv \neg \Diamond_t \mathcal{B}_{\text{n}} \neg p \wedge \Diamond_t \mathcal{B}_{\text{n}} \neg p$. The right-hand side is of the form $\neg a \wedge a$, which must be False. This contradicts our initial assumption that $\mathcal{B}_*$ is t-deceptive (i.e., $\Diamond_t \mathcal{B}_* p \wedge \Diamond_t \mathcal{B}_* \neg p$ is True). Therefore, $\mathcal{B}_*$ is t-non-deceptive.

B.5.2 Theoretically Validating Defenses to Hyperparameter Deception

We prove Theorem 1:

Theorem 1. *Suppose that the set of allowable hyper-HPs C of H is constrained, such that any two allowable random-search distributions μ and ν have Renyi-∞-divergence at most a constant, i.e. $D_\infty(\mu||\nu) \leq \gamma$. The (K, R)-defended random-search EHPO of Definition 7 is guaranteed to be t-non-deceptive if we set $R \geq \sqrt{t \exp(\gamma K)/K} = O(\sqrt{t})$.*

Suppose we are considering HPO via random search [54], in which the set of allowable hyper-hyperparameters contains tuples (μ, M), where μ is a distribution over all possible hyperparameter sets Λ and M is the number of different

hyperparameter configuration trials to run. This set S is the Cartesian product of the set of allowable distributions D ($\mu \in D$) and M.

Suppose that for any two allowable distributions $\mu, \nu \in D$ and any event A (a measurable subset of Λ), $\mu(A) \leq e^{\gamma} \cdot \nu(A)$ (i.e., the Renyi ∞-divergence between any pair of distributions is at most γ). This bounds how much the choice of hyper-hyperparameter can affect the hyperparameters in HPO.

We also suppose we start from a naive reasoner (expressed via the operator $\mathcal{B}_{\text{n}}$), which draws conclusions based on a log with K trials. For this scenario, we are only concerned with the reasoner's conclusions from K-trial logs. We therefore assume w.l.o.g. that the reasoner draws no conclusions unless presented with exactly one log with exactly K trials.

For some constant $R \in \mathbb{N}$, we construct a new reasoner $\mathcal{B}_*$ that does the following: It draws conclusions only from a single log with exactly KR trials (otherwise it concludes nothing). To evaluate a proposition p, it splits the log into R groups of K trials each, evaluates $\mathcal{B}_n$ on p on each of those R groups, and then concludes p only if $\mathcal{B}_n$ also concluded p on all R groups.

Now consider a particular (arbitrary) proposition p. Since $\mathcal{B}_*$ draws conclusions based on only a single log, any strategy σ for the demon is equivalent to one that maintains at most one log at all times (the "best" log it found so far for its purposes, as it can discard the rest).

Let Q be the supremum, taken over all allowable distributions μ, of the probability that running a group of K random search trials using that distribution will result in a log that would convince the $\mathcal{B}_n$ of p. Similarly, let $Q_\neg$ be the supremum, taken over all allowable distributions ν, of the probability that run-

ning a group of K trials using that distribution will result in a log that would convince $\mathcal{B}_n$ of $\neg p$.

Observe that Q is the probability of an event in a product distribution of K independent random variables each distributed according to μ, and similarly for $Q_\neg$, and the corresponding events are disjoint. By independent additivity of the Renyi divergence, the Renyi ∞-divergence between these corresponding product measures will be γK. It follows that

$$1 - Q \geq \exp(-\gamma K) Q_\neg$$

and

$$1 - Q_\neg \geq \exp(-\gamma K) Q$$

From here it's fairly easy to conclude that

$$Q + Q_\neg \leq \frac{2}{1 + \exp(-\gamma K)}.$$

Now, an EHPO procedure using random search with KR trials will convince $\mathcal{B}_*$ of p with probability Q^R, since all R independently sampled groups of K trials must "hit" and each hit happens with probability Q. Thus, the expected time it will take the fastest strategy to convince us of p is $Q^{-R} \cdot KR$. Similarly, the fastest strategy to convince us of $\neg p$ takes expected time $Q_\neg^{-R} \cdot KR$.

Suppose now, by way of contradiction, that the t-non-deceptiveness axiom is violated, and there are strategies that can achieve both of these in time at most t. That is,

$$Q^{-R} \cdot KR \leq t \qquad \text{and} \qquad Q_\neg^{-R} \cdot KR \leq t.$$

From here, it's fairly easy to conclude that

$$Q + Q_\neg \geq 2\left(\frac{KR}{t}\right)^{1/R}.$$

Combining this with our conclusion above gives

$$\left(\frac{KR}{t}\right)^{1/R} \leq \frac{1}{1+\exp(-\gamma K)}.$$

It's clear that we can cause this to be violated by setting R to be large enough. Observe that

$$\frac{1}{1+\exp(-\gamma K)} \leq \exp(-\exp(-\gamma K)),$$

so

$$\left(\frac{KR}{t}\right)^{1/R} \leq \exp(-\exp(-\gamma K)).$$

Taking the root of both sides gives

$$\left(\frac{KR}{t}\right)^{\frac{1}{R\exp(-\gamma K)}} \leq \frac{1}{e}.$$

To simplify this expression, let β denote

$$\beta = R\exp(-\gamma K).$$

So that

$$\left(\frac{\beta K}{t\exp(-\gamma K)}\right)^{1/\beta} \leq \frac{1}{e}.$$

Finally, we set R such that

$$\beta = \sqrt{\frac{t\exp(-\gamma K)}{K}}.$$

To give

$$\left(\frac{1}{\beta}\right)^{1/\beta} \leq \frac{1}{e}.$$

But this is impossible, as the minimum of x^x occurs above $1/e$. This setting of R gives

$$R = \beta\exp(\gamma K) = \sqrt{\frac{t\exp(\gamma K)}{K}} = O(\sqrt{t}).$$

This shows that, for this task, if we run our constructed EHPO with $R = O(\sqrt{t})$ assigned in this way, it will be guaranteed to be t-non-deceptive.

B.5.3 Defense Experiments

In this section we provide more information about the implementation of a random-search-based defense to hyperparameter deception in Wilson et al. [623], which we discuss in Section 2.5.

Our Implemented Defense Algorithm

The defense we implement in our experiments is a bit different than what we describe in our theoretical results in Section 2.5. In particular, in practice it is easier to implement subsampling rather than resampling.

Protocol of Selecting Hyper-HPs. As partially illustrated in Figure 2.1 and elaborated on in Appendix B.2, Wilson et al. [623]'s choice of hyper-HPs does not capture the space where Adam effectively simulates Heavy Ball. In Wilson et al. [623]i, Adam-specific HPs like numerical variable $\epsilon = 10^{-8}$ [314] are treated as constants, leading to a biased HP-search space.

In contrast, we select the hyper-HPs of ϵ following a dynamic searching protocol:

Inspired by Choi et al. [122], we start from a wide range $\epsilon \in [10^{-12}, 10^{12}]$ as a wide search space. We iteratively select powers-of-10 grids that are uniformly spaced in the logarithmic scale of the current range. For instance, the selected grids for the prior range would be $\{10^{-12}, 10^{-11}, \cdots, 10^{11}, 10^{12}\}$. We perform a single run on each grid selected, and shrink the range towards grids where the best performance are achieved. The shrinkage follows the policy of either $\times 10$ to the lower boundary or $\times 0.1$ to the upper boundary. For example, for the prior

range, we found the best performance is achieved on grid 10^{11}, so we multiply the lower boundary 10^{-12} with 10 and shift the range to $\epsilon \in [10^{-11}, 10^{12}]$. Our protocol terminates with $\epsilon \in [10^{10}, 10^{12}]$ as the final hyper-HPs that we use for our defended random search EHPO.

Scaling Learning Rate η. Note that directly applying the hyper-HP of $\epsilon \in [10^{10}, 10^{12}]$ to Adam would lead to extremely slow convergence, since essentially large ϵ indicates a small effective learning rate η. Similar to Choi et al. [122], we explore a shifted hyper-HPs for the η, scaled proportionally with ϵ. Specifically, note that a large ϵ would make the update of Adam approach the update rule in the Heavy Ball method; for any randomly selected $\epsilon \in [10^{10}, 10^{12}]$, we perform the random search of η/ϵ instead of η itself in the space of $[0.5, 0.7]$, which is the search space of HB's learning rate shown in Wilson et al. [623].

Experimental setup

We follow the setup from [623], where the details are specified in Section B.2.2.

Code

The code for running these experiments can be found at `https://github.com/pasta41/deception`.

Associated results and logs

In line with our notion of a log, we provide heatmaps of our logs in Figures B.12, B.13 and that correspond with our results in Section 2.5. We note that the per-

formance of Heavy Ball for some random seeds is very bad (e.g., 10% test accuracy). The performance varies widely – also nearing 92% for different random seeds. We affirm that this is the search space that yields the best results for Heavy Ball (92%).

The results for Heavy Ball exhibit large variance. This illustrates a strength of our defensed: it actually helps with robustness against potentially making the wrong conclusion about Heavy Ball's performance (more generally), due to not making conclusions off of a single result (and perhaps using a random seed for which performance is particularly bad). We make a different claim about relative algorithm performance than Wilson et al. [623] about Heavy Ball (i.e., we do not claim that it is better than Adam); but we do not reach this conclusion for the wrong reason (i.e., that we got one bad Heavy Ball result for a particular random seed).

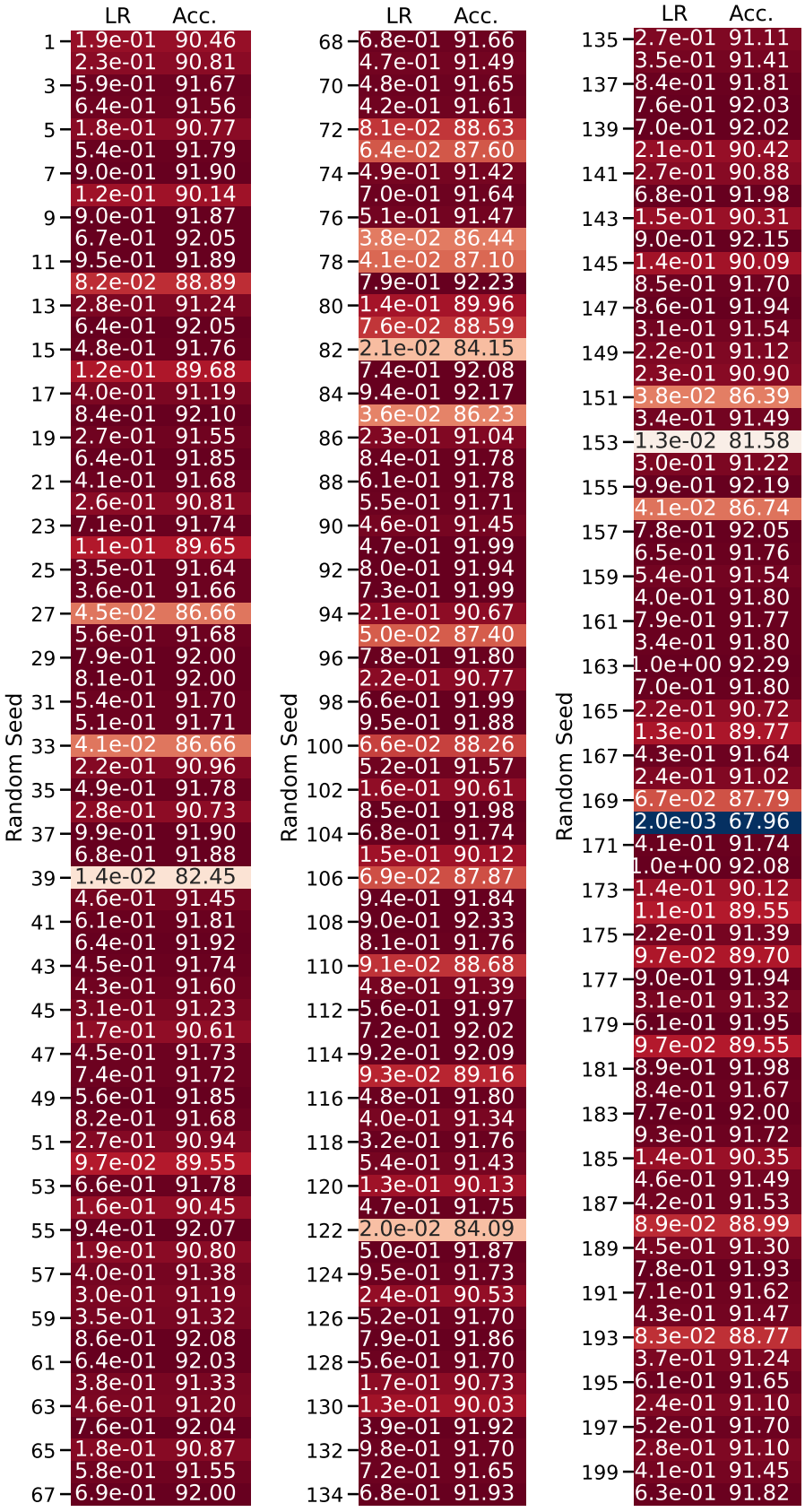


Figure B.9: Heatmap logs of SGD defended random search. Redder rows indicate higher test accuracy.

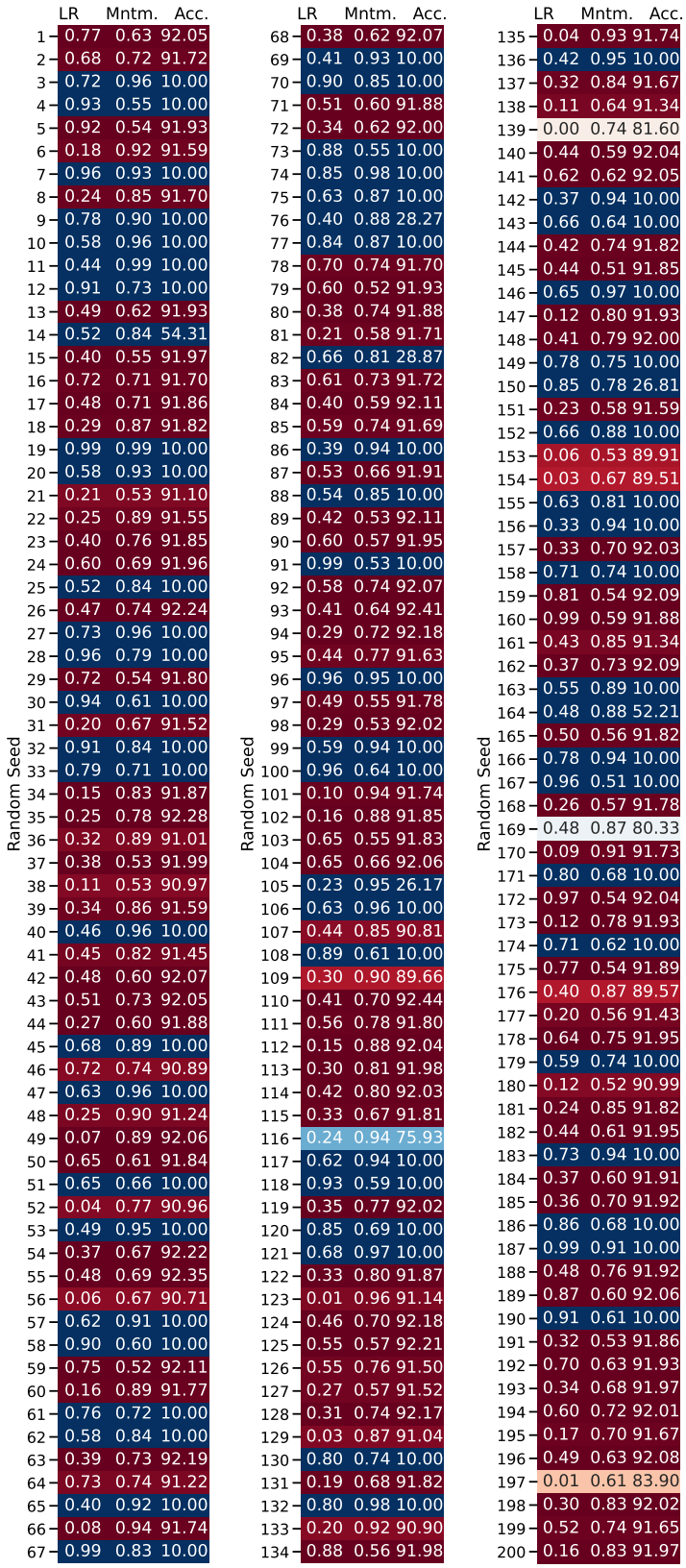


Figure B.10: Heatmap logs of Heavy Ball (HB) defended random search. Redder rows indicate higher test accuracy.

Figure B.11: Heatmap logs of Adam defended random search. Redder rows indicate higher test accuracy.

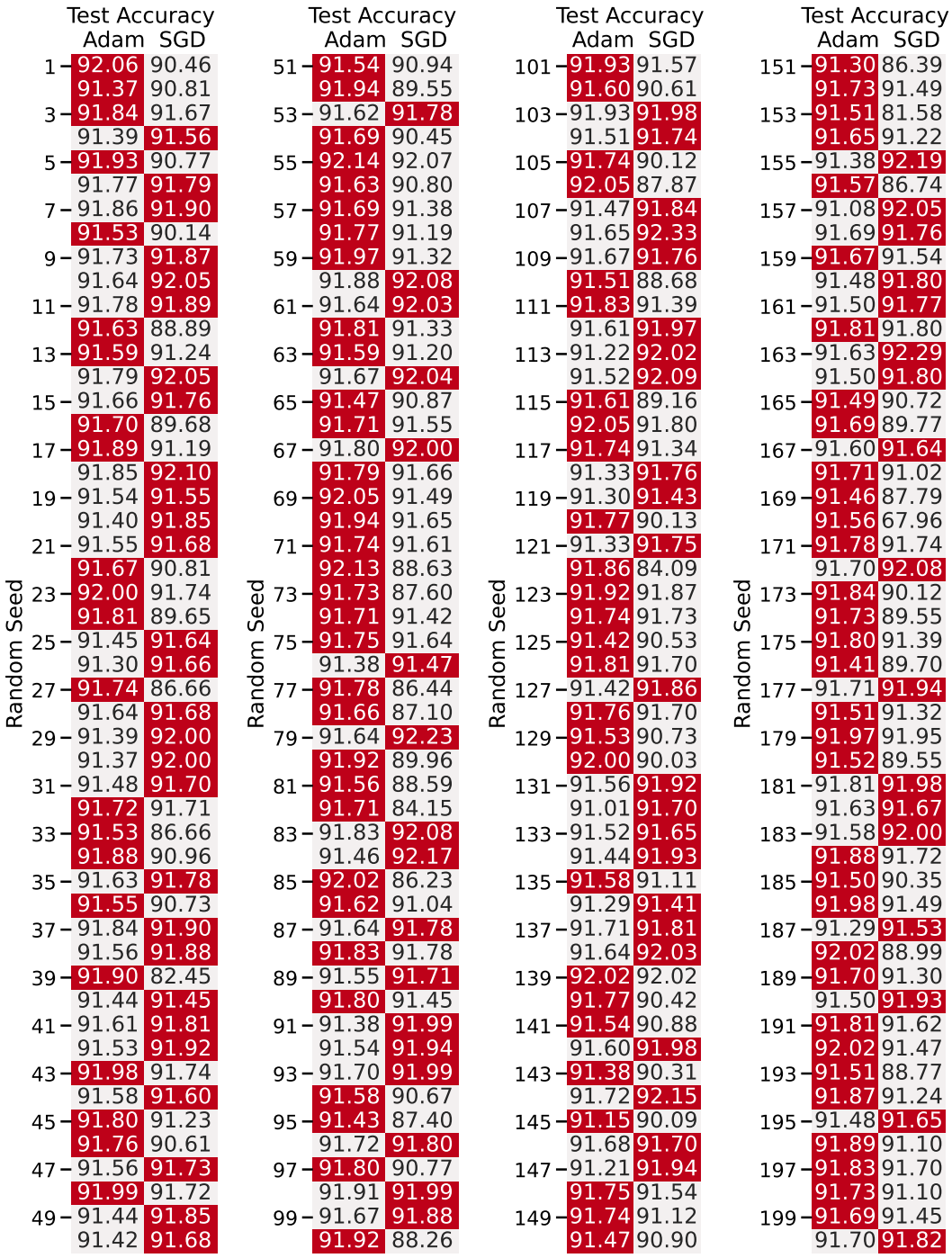


Figure B.12: Heatmap logs of test accuracy of VGG-16 on CIFAR-10 for Adam vs. SGD using our defended random search EHPO. Red indicates higher test accuracy for the given random seed.

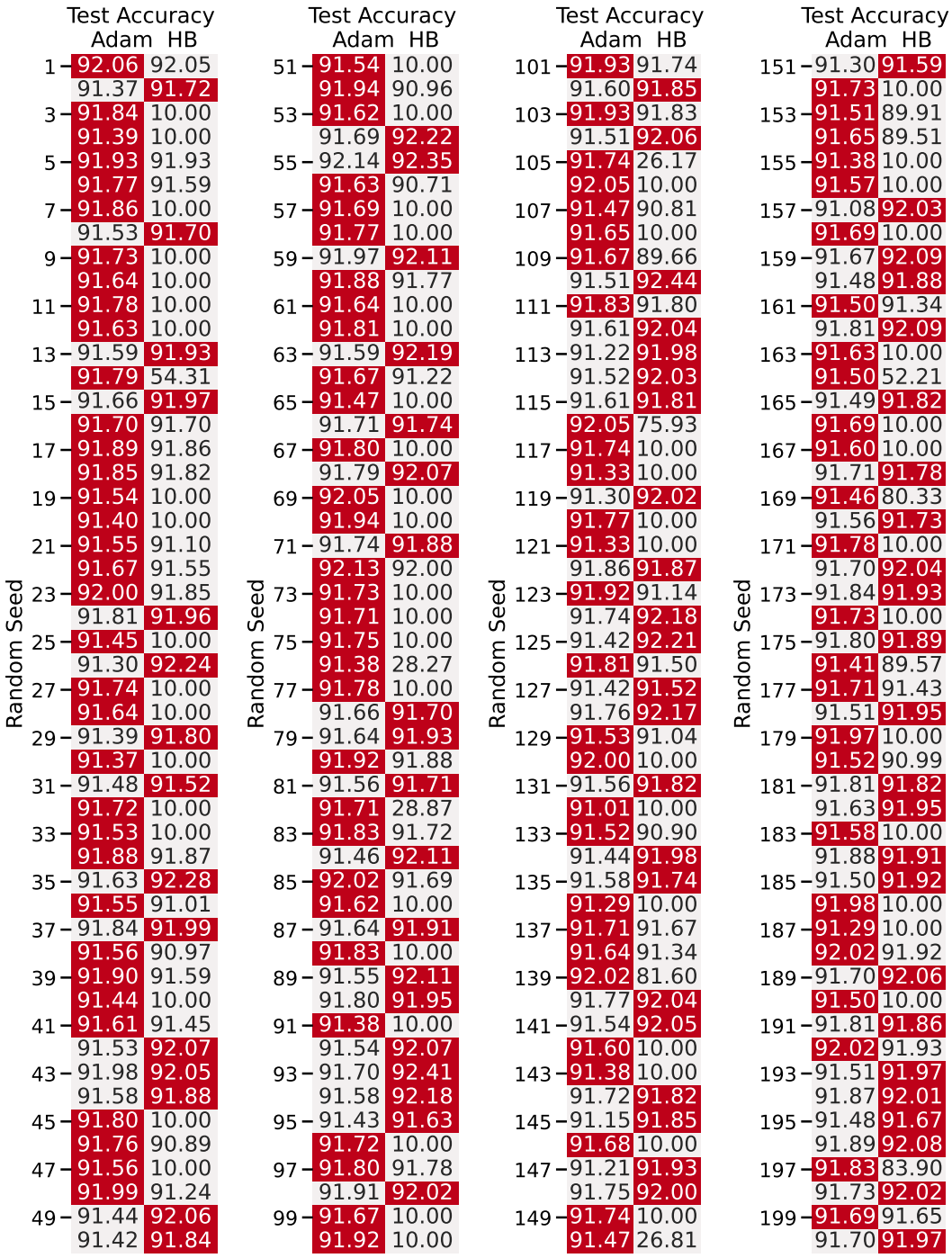


Figure B.13: Heatmap logs of test accuracy of VGG-16 on CIFAR-10 for Adam vs. Heavy Ball (HB) using our defended random search EHPO. Red indicates higher test accuracy for the given random seed.

B.6 Section 2.6 Appendix: Notes on Conclusion

B.6.1 Additional Practical Takeaways

In our conclusion in Section 2.6, we note the following practical takeaways:

- **Researchers should have their own notion of skepticism, appropriate to their specific task.** There is no one-size-fits-all defense solution. Our results are ***broad insights*** about defended EHPO: A defended EHPO is ***always possible***, but finding an efficient one will depend on the task.
- **Researchers should make explicit how they choose hyper-HPs.** What is reasonable is ultimately a function of what the ML community accepts. Being explicit, rather than eliding hyper-HP choices, is essential for helping decide what is reasonable. As a heuristic, we recommend setting hyper-HPs such that they include HPs for which the optimizers' performance starts to degrade, as we do above.
- **Avoiding hyperparameter deception is just as important as reproducibility**. We have shown that reproducibility [80, 249, 271, 470, 542] is only part of the story for ensuring reliability. While necessary for guarding against brittle findings, it is not sufficient. We can replicate results—even statistically significant ones—that suggest conclusions that are altogether wrong.

We elaborate here that our defended random search EHPO indicates a particular form of skepticism that may (or may not) be appropriate to different ML tasks. That is, we suggest a defended EHPO, but do not claim that that EHPO

is optimal or suited for all tasks. Even though it may not be optimal, the guarantees it affords would translate to other tasks (so long as the assumption is maintained that there is an upper bound on how much the hyper-HPs can control the HPs). So, while we do not necessarily encourage practitioners to use our particular defended EHPO, we do not discourage it either. The main take away is that practitioners should develop their own notion of skepticism (appropriate to their particular task) and be explicit about the assumptions they rely on when selecting hyper-HPs. The way one chooses hyper-HPs should be defensible.

When in doubt, as a heuristic we recommend using a search space that includes where an algorithm's performance starts to degrade (to be assured that a maximum, even if a local one, has been found). We refer to our dynamic two-phase protocol (which we describe in detail in Appendix B.5.3) as an example of how to do this. We first do a broad (but coarse) search. We used grid search for that initial sweep. Random search may be a better choice for some tasks. We were familiar with Wilson et al. [623] (and many have written about it), and felt confident that grid search would capture the space well based on the results that others have also reported on this task. We then used this first sweep to determine which hyper-HPs we should use for our second, finer-grained sweep. We apply our more expensive, defended EHPO for this second sweep, using the hyper-HPs we selected from the first sweep. In other words, we spent a bit of time/our compute budget justifying to ourselves that we were picking reasonable hyperparameters – instead of just picking one grid or range for random search to sample, and hoping that our results would be representative of other search spaces.

B.6.2 Broader Impact

As we suggest in Section 2.5, our work can be considered as related to (but orthogonal with) with prior studies on reproducibility as advocating for more robust scientific practices in ML research. In particular, our work complements prior empirical studies that shine a light on reliability issues in ML—issues that relate particularly to traditionally underspecified choices in hyperparameter optimization [122, 376, 543]. In contrast to this prior work, which illustrates the issue with experiments, we provide a theoretical contribution that enables ML practitioners and researchers to defend against unreliable, inconsistent HPO. We provide a theoretically-backed mechanism to promote and facilitate more trustworthy norms and practices in ML research.

More broadly, our work can be understood as a mechanism for dealing with *measurement bias*—the misalignment between what one intends to measure and what they are actually measuring—for overall ML algorithm performance. While alleviating measurement bias is by no means novel to more mature branches of science [243], including other fields of computing [431], until recently it has been under-explored in ML. Beginning in the last couple of years, measurement bias is now coming under increased scrutiny with respect to the origins of empirical gains in ML [210, 430]. In current work, it is often difficult to disentangle whether the concluded measured performance gains are due to properties of the training algorithm or to fortuitous HP selection. Our formalization, rather than allowing HPO choices to potentially obscure empirical results, provides confidence in the conclusions we can draw about overall algorithm performance.

Our work also highlights how there is a human element, not a just statisti-

cal one, to bias in ML pipelines: Practitioners make decisions about HPO that can heavily influence performance (e.g., choice of hyper-hyperparameters). The human element of biasing solution spaces has been discussed in sociotechnical writing [136, 215, 547], in AI [417], in the context of "p-hacking" or "fishing expeditions" for results that fit a desired pattern [227], and was also the focus of Professor Charles Isbell's NeurIPS 2020 keynote [295]. Formalizing the process for how to draw conclusions from HPO, as we do here, has the potential to alleviate the effects of this type of human bias in ML systems.

Lastly, our insights concerning robustness also extend to growing areas in ML that use learning to guide hyperparemeter selection, such as meta-learning and neural architecture search [96, 189, 288, 289, 649]. While the assisting learning agents in those methods guide choosing hyperparameters for the trained output model, their own hyperparameters tend to be either manually tuned or chosen with more traditional HPO strategies, like grid search [647]. In other words, these processes can exhibit the bias problem discussed above and are therefore potentially subject to hyperparameter deception, which can be mitigated by the work we present here.

APPENDIX C

APPENDIX FOR ARBITRARINESS AND SOCIAL PREDICTION

The full paper's appendix is extensive, containing figures for every experiment. We provide an abridged version here, and defer the arXiv version of the paper for the full set of results.

C.1 Extended Preliminaries

C.1.1 Notes on notation and on our choice of terminology

Terminology. Traditionally, what we term "observed labels" o are often referred to instead as the "ground truth" or "correct" labels [7, 262, 325, e.g.]. We avoid this terminology because, as the work on label bias has explained, these labels are often unreliable or contested [136, 212].

Sets, random variables, and instances. We use bold non-italics letters to denote random variables (e.g., $\mathbf{x}$, $\mathbf{D}$), capital block letters to denote sets (e.g., $\mathbb{X}$, $\mathbb{Y}$), lower case italics letters to denote scalars (e.g., o), bold italics lower case letters to denote vectors (e.g., $\boldsymbol{x}$), and bold italics upper case to denote matrices (e.g., $\boldsymbol{D}_k$). For a complete example, $\boldsymbol{x}$ is an arbitrary instance's feature vector, $\mathbb{X}$ is the set representing the space of instances $\boldsymbol{x}$ ($\boldsymbol{x} \in \mathbb{X}$), and $\mathbf{x}$ is the random variable that can take on specific values of $\boldsymbol{x} \in \mathbb{X}$. We use this notation consistently, and thus do not always define all symbols explicitly.

C.1.2 Constraints on our setup

Our setup, per our definition of the learning process (Definition 8) is deliberately limited to studying the effects of variance due to changes in the underlying training dataset, with such datasets drawn from the same distribution. For this reason, Definition 8 does not include the data collection process or hyperparameter optimization (HPO), which can further introduce non-determinism to machine learning, and are thus assumed to have been already be completed.

Relatedly, variance-induced error can of course have other sources due to such non-determinism. For example, stochastic optimization methods, such as SGD and Adam, can cause fluctuations in test error; as, too, can choices in HPO configurations [145]. While each of these decision points is worthy of investigation with respect to their impact on fair classification outcomes, we aim to fix as many sources of randomness as possible in order to highlight the particular kind of arbitrariness that we describe in Sections 3.1 and 3.3. As such, we use the Limited-memory BFGS solver and fix our hyperparameters based on the results of an initial search (Section 3.5), for which we selected a search space through consulting related work such as Chen et al. [119].

C.1.3 Costs and the classification decision threshold

For reference, we provide a bit more of the basic background regarding the relationship between the classification decision threshold τ and costs of false positives `FP` (C_{01}) and false negatives `FN` (C_{10}). We visualize the loss as follows:

Table C.1: Confusion matrix for cost-sensitive loss ℓ, adapted from Elkan [188].

	$\hat{y} = 0$	$\hat{y} = 1$
$o = 0$	TN: 0	FP: C_{01}
$o = 1$	FN: C_{10}	TP: 0

0-1 loss treats the cost of different types of errors equally $C_{01} = C_{10} = 1$); false positives and false negatives are quantified as equivalently bad – they are *symmetric*; the case for which $C_{01} \neq C_{10}$ is *asymmetric* or *cost-sensitive*.

Altering the asymmetric of costs shifts the classification decision threshold τ applied to the underlying regressor $r_{\boldsymbol{D}_k}$. We can see this by examining the behavior of $r_{\boldsymbol{D}_k}$ that we learn. $r_{\boldsymbol{D}_k}$ estimates the probability of a each label given $\boldsymbol{x}$ (since we do not learn using $\boldsymbol{g}$), i.e., that we develop a good approximation of the distribution $p(\mathbf{y}|\mathbf{x})$. Ideally, $r_{\boldsymbol{D}_k}$ will be similar to the Bayes optimal classifier (for which the classification rule produces classifications y^* that yield the smallest weighted sum of the loss, where the weights are the probabilities of a particular label $\mathbf{y} = i$ for a given $(\boldsymbol{x}, \boldsymbol{g})$, i.e., sums over

$$p(\mathbf{y} = i|\mathbf{x} = \boldsymbol{x})\ \ell(i, y'). \tag{C.1}$$

For binary classification, the terms of (C.1) in the sum for a particular y' yield two cases:

- $i = y'$: By definition, $\ell(i, y') = 0$; therefore, (C.1) = 0.
- $i \neq y'$: By definition, $\ell(i, y') = C_{01}$ or $\ell(i, y') = C_{10}$. So, (C.1) will weight the cost by the probability $p(\mathbf{y} = i|\mathbf{x} = \boldsymbol{x})$.

We can therefore break down the Bayes optimal classifier into the following

decision rule, which we hope to approximate through learning. For an arbitrary $(\boldsymbol{x}, \boldsymbol{g})$ and $\mathbb{Y} = \{0, 1\}$,

$$\min\Big(\overbrace{\overbrace{p(\mathbf{y}=0|\mathbf{x}=\boldsymbol{x})}^{\text{Probability of FP}}\times C_{01} + \overbrace{p(\mathbf{y}=1|\mathbf{x}=\boldsymbol{x})}^{\text{Probability of TP}}\times 0}^{\text{Weighted cost of predicting positive (1) class}},\ \overbrace{\overbrace{p(\mathbf{y}=0|\mathbf{x}=\boldsymbol{x})}^{\text{Probability of TN}}\times 0 + \overbrace{p(\mathbf{y}=1|\mathbf{x}=\boldsymbol{x})}^{\text{Probability of FN}}\times C_{10}}^{\text{Weighted cost of predicting negative (0) class}}\Big)$$
$$= \min\Big(\overbrace{p(\mathbf{y}=0|\mathbf{x}=\boldsymbol{x})}^{\text{Probability of FP}}\times C_{10},\ \overbrace{p(\mathbf{y}=1|\mathbf{x}=\boldsymbol{x})}^{\text{Probability of FN}}\times C_{10}\Big)$$

That is, to predict label 1, the cost of mis-predicting 1 (i.e., the cost of a false positive FP) must be be smaller than the cost of mis-predicting 0 (i.e, the cost of a false negative FN). In binary classification $p(\mathbf{y}|\mathbf{x} = \boldsymbol{x}) = p(\mathbf{y} = 1|\mathbf{x} = \boldsymbol{x}) + p(\mathbf{y} = 0|\mathbf{x} = \boldsymbol{x}) = 1$. So, we can assign $p(\mathbf{y} = 1|\mathbf{x} = \boldsymbol{x}) = \tau$ and $p(\mathbf{y} = 0|\mathbf{x} = \boldsymbol{x}) = 1 - \tau$, and rewrite the above as

$$\min\Big((1-\tau)\, C_{01},\ \ \tau C_{10}\Big). \tag{C.2}$$

The decision boundary is the case for which both of the arguments to min in (C.2) are equivalent (i.e., the costs of predicting a false positive and a false negative are equal), i.e.,

$$(1-\tau)\, C_{01} = \tau C_{10} \Rightarrow \tau = \frac{C_{01}}{C_{01} + C_{10}},\ \text{so,}$$
$$h_{\boldsymbol{D}_k}(\boldsymbol{x}) = \mathbf{1}[r_{\boldsymbol{D}_k}(\boldsymbol{x}) \geq \tau] = \begin{cases} 1, & \text{if } p(\mathbf{y}=1|\mathbf{x}=\boldsymbol{x}) \geq \tau = \frac{C_{01}}{C_{01}+C_{10}} \\ 0, & \text{otherwise.} \end{cases}$$

For 0-1 loss, in which $C_{01} = C_{10} = 1$, τ evaluates to $\frac{1}{2}$. If we want to model asymmetric costs, then we need to change this decision threshold to account for which type of error is more costly. For example, let us say that false negatives

are more costly than false positives, with $C_{01} = 1$ and $C_{10} = 3$. This leads to a threshold of $\frac{1}{4}$, which biases $h_{\boldsymbol{D}_k}$ toward choosing the (generally cheaper to predict/more conservative) positive class.

C.1.4 The bootstrap method

In the bootstrap method, we treat each dataset $\hat{\boldsymbol{D}}_k \in \hat{\mathbb{D}}$ as equally likely. For each set aside test example $(\boldsymbol{x}, \boldsymbol{g}, o)$, we can approximate $\texttt{Err}(\mathcal{A}, \mathbb{D}, (\boldsymbol{x}, \boldsymbol{g}, o))$ empirically by computing

$$\hat{\texttt{Err}}(\mathcal{A}, \hat{\mathbb{D}}, (\boldsymbol{x}, \boldsymbol{g}, o)) = \frac{1}{B} \sum_{i=1}^{B} \ell(o, \hat{h}_{\hat{\boldsymbol{D}}_i}(\boldsymbol{x})) \tag{C.3}$$

for a concrete number of replicates B. We estimate overall error $\hat{\texttt{Err}}(\mathcal{A}, \hat{\mathbb{D}})$ for the test set by additionally summing over each example instance $(\boldsymbol{x}, \boldsymbol{g}, o)$, which we can further delineate into $\hat{\texttt{FPR}}$ and $\hat{\texttt{FNR}}$, or into group-specific $\hat{\texttt{Err}}_{\boldsymbol{g}}$, $\hat{\texttt{FPR}}_{\boldsymbol{g}}$, and $\hat{\texttt{FNR}}_{\boldsymbol{g}}$ by computing separate averages according to $\boldsymbol{g}$.

The bootstrap method exhibits less variance than cross-validation, but can be biased — in particular, pessimistic — with respect to estimating expected error. To reduce this bias, one can follow our setup in Definition 8, which splits into train and test sets before resampling. For more information comparing the two methods, see Efron and Tibshirani [185, 186]. Further, recent work shows that, in relation to studying individual models, CV is in fact asymptotically uninformative regarding expected error [608].

C.2 Additional Details on Variance and Self-Consistency

In this appendix, we provide more details on other types of statistical error (Appendix C.2.1), on variance (Appendix C.2.2) and self-consistency (Appendix C.2.3). Following this longer presentation of our metrics, we then provide some additional information on other definitions of variance that have been used in work on fair classification, and contextualize issues with these definitions that encouraged us to deviate from them in order to derive our definition of self-consistency (Appendix C.3).

C.2.1 Other statistical sources of error

Noise. Noise is traditionally understood as *irreducible error*; it is due to inherent randomness in the data, which cannot be captured perfectly accurately by a deterministic decision rule $h_{\boldsymbol{D}_k}$. Notably, noise is an aspect of the data collection pipeline, not the learning process (Definition 8). It is *irreducible* in the sense that it does not depend on our choice of training procedure $\mathcal{A}$ or how we draw datasets for training from $\mathbb{D}$, either in theory or in practice. Heteroskedastic noise across demographic groups is often hypothesized to be a source of unfairness in machine learning [119, 136]. Importantly, albeit somewhat confusingly, this is commonly referred to as label bias, where "bias" connotes discrimination, as opposed to the statistical bias that we mention here.

Unlike noise, bias and variance are traditionally understood as sources of epistemic uncertainty. These sources of error are *reducible* because they are contingent on the modeling choices we make in the learning process; if we knew

how to model the task at hand more effectively, in principle, we could reduce bias and variance error.

Bias. Within the amount of reducible error, bias reflects the error associated with the chosen hypothesis class $\mathbb{H}$, and is therefore governed by decisions concerning the training procedure $\mathcal{A}$ in the learning process (Definition 8). This type of error is persistent because it takes effect at the level of possible models in $\mathbb{H}$; in expectation, all models $h_{\boldsymbol{D}_k} \in \mathbb{H}$ have the same amount of bias-induced error.

Whereas variance depends on stochasticity in the underlying training data, noise and bias error are traditionally formulated in relation to the Bayes optimal classifier — the best possible classifier that machine learning could produce for a given task [7, 178, 229]. Since the Bayes optimal classifier is typically not available in practice, we often cannot estimate noise or bias directly in experiments.

Of the three types of statistical error, it is only variance that seems to reflect the intuition in Figure 3.1 concerning the behavior of different possible models $h_{\boldsymbol{D}_k}$. This is because noise is a property of the data distribution; for a learning process (Definition 8), in expectation we can treat noise error as constant. Bias can similarly be treated as constant for the learning process: It is a property of the chosen hypothesis class $\mathbb{H}$, and thus is in expectation the same for each $h_{\boldsymbol{D}_k} \in \mathbb{H}$. In Figure 3.1, we are keeping the data distribution constant and $\mathbb{H}$ constant; we are only changing the underlying subset of training data to produce different models $h_{\boldsymbol{D}_k}$.

C.2.2 Our variance definition

We first provide a simple proof that explains the simplified version for our empirical approximation for variance in (3.1).

Proof. For the models $\{h_{\boldsymbol{D}_b}\}_{b=1}^{B}$ that we produce, we denote $\hat{\mathbb{Y}}$ to be the multiset of their predictions on $(\boldsymbol{x}, \boldsymbol{g})$. $|\hat{\mathbb{Y}}| = B = B_0 + B_1$, where B_0 and B_1 represent the counts of 0 and 1-predictions, respectively. We also set the cost of false positives to be $\ell(0,1) = C_{01}$ and the cost of false negatives to be $\ell(1,0) = C_{10}$.

Looking at the sum in $\hat{\texttt{var}}$ (i.e., $\sum_{i \neq j}$), each of the B_0 0-predictions will get compared to the other $B_0 - 1$ 0-predictions and to the B_1 1-predictions. By the definition of ℓ, each of the $B_0 - 1$ computations of $\ell(0,0)$ evaluates to 0 and each of the B_1 computations of $\ell(0,1)$ evaluates to C_{01}. Therefore, the B_0 0-predictions contribute

$$B_0 \times [(0 \times (B_0 - 1)) + C_{01} \times B_1] = C_{01} B_0 B_1$$

to the sum in $\hat{\texttt{var}}$, and, by similar reasoning, $B_1 \times [(0 \times (B_1 - 1)) + C_{10} \times B_0] = C_{10} B_0 B_1$. It follows that the total sum in $\hat{\texttt{var}}$ is

$$\sum_{i \neq j} \ell\Big(\hat{h}_{\hat{\boldsymbol{D}}_i}(\boldsymbol{x}), \hat{h}_{\hat{\boldsymbol{D}}_j}(\boldsymbol{x})\Big) = (C_{01} + C_{10}) B_0 B_1. \text{ Therefore}$$

$$\overbrace{\frac{1}{B(B-1)} \sum_{i \neq j} \ell\Big(\hat{h}_{\hat{\boldsymbol{D}}_i}(\boldsymbol{x}), \hat{h}_{\hat{\boldsymbol{D}}_j}(\boldsymbol{x})\Big)}^{\hat{\texttt{var}}\left(\mathcal{A}, \hat{\mathbb{D}}, (\boldsymbol{x}, \boldsymbol{g})\right)} = \overbrace{\frac{(C_{01} + C_{10}) B_0 B_1}{B(B-1)}}^{(3.1)}$$

□

The effect of τ on variance. As discussed in Appendix C.1.3, C_{01} and C_{10} can be related to changing τ applied to $r_{\boldsymbol{D}_k}$ to produce classifier $h_{\boldsymbol{D}_k}$. We analyze the

range of minimal and maximal empirical variance by examining the behavior of $B \to \infty$, i.e.,

$$\lim_{B\to\infty} \frac{(C_{01} + C_{10})B_0B_1}{B(B-1)}. \tag{C.4}$$

Minimal variance. Either B_0 or B_1 (exclusively, since $B_0 + B_1 > 1$) will be $= 0$, with the other being $= B$, making (C.4) equivalent to

$$\lim_{B\to\infty} \frac{(C_{01} + C_{10}) \times 0}{B(B-1)} = 0, \text{regardless of the value of } C_{01} + C_{10}.$$

Maximal variance. B_0 will represent half of B, with B_1 representing the other half. More particularly, $B_0 = \frac{B}{2}$ and $B_1 = \frac{B}{2}$; or, without loss of generality, $B_0 = \frac{B-1}{2}$ and $B_1 = \frac{B+1}{2}$. This means that

$$\begin{aligned}
\frac{(C_{01} + C_{10})B_0B_1}{B(B-1)} &= \frac{(C_{01} + C_{10})(\frac{B}{2})^2}{B(B-1)} && \left(\text{Or, } = \frac{(C_{01} + C_{10})(\frac{B-1}{2})(\frac{B+1}{2})}{B(B-1)}\right) \\
&= \frac{(C_{01} + C_{10})(\frac{B^2}{4})}{B^2 - B} && \left(\text{Or, } = \frac{(C_{01} + C_{10})(\frac{(B^2-1}{4})}{B(B-1)}\text{; it will not matter in the limit}\right) \\
&= \frac{(C_{01} + C_{10})B^2}{4B^2 - 4B}.
\end{aligned}$$

And, therefore,

$$\lim_{B\to\infty} \frac{(C_{01} + C_{10})B^2}{4B^2 - 4B} = \frac{C_{01} + C_{10}}{4}. \tag{C.5}$$

It follows analytically that variance will be in the range $[0, \frac{C_{01}+C_{10}}{4})$. However, empirically, for concrete B, $\hat{\mathtt{var}}(\mathcal{A}, \hat{\mathbb{D}}, (\boldsymbol{x}, \boldsymbol{g})) \to [0, \frac{C_{01}+C_{10}}{4} + \epsilon]$, for smaller positive ϵ as the number of models B increases. The maximal variance will better approximate $\frac{C_{01}+C_{10}}{4}$ as B gets larger, but will be $> \frac{C_{01}+C_{10}}{4}$. For example, for 0-1 loss $\frac{C_{01}+C_{10}}{4} = \frac{2}{4} = 0.5$. For $B = 100$, the maximal $\hat{\mathtt{var}}(\mathcal{A}, \hat{\mathbb{D}}, (\boldsymbol{x}, \boldsymbol{g})) = \frac{2\times50\times50}{100\times99} = \frac{50}{99} \approx .505$.

C.2.3 Deriving self-consistency from variance

In this appendix, we describe the relationship between variance (Definition 9) and self-consistency (Definition 10) in more detail, and show that $\hat{\mathtt{SC}}(\mathcal{A}, \{\boldsymbol{D}_b\}_{b=1}^B, (\boldsymbol{x}, \boldsymbol{g})) \to [0.5 - \epsilon, 1]$ for small positive ϵ as the number of models B increases.

Proof. Note that, by the definition of 0-1 loss, $C_{01} = C_{10} = 1$, so

$$\hat{\mathtt{var}}(\mathcal{A}, \hat{\mathbb{D}}, (\boldsymbol{x}, \boldsymbol{g}))_{0\text{-}1} = \frac{1}{B(B-1)} \sum_{i \neq j} \mathbf{1}[h_{\boldsymbol{D}_i}(\boldsymbol{x}) \neq h_{\boldsymbol{D}_j}(\boldsymbol{x})] = \frac{2B_0B_1}{B(B-1)}. \tag{C.6}$$

By the definition of the indicator function $\mathbf{1}$,

$$\begin{aligned} 1 &= \frac{1}{B(B-1)} \sum_{i \neq j} \Big[\overbrace{\mathbf{1}[h_{\boldsymbol{D}_i}(\boldsymbol{x}) \neq h_{\boldsymbol{D}_j}(\boldsymbol{x})]}^{\text{From } \hat{\mathtt{var}}\left(\mathcal{A}, \hat{\mathbb{D}}, (\boldsymbol{x}, \boldsymbol{g})\right)_{0\text{-}1}} + \overbrace{\mathbf{1}[h_{\boldsymbol{D}_i}(\boldsymbol{x}) = h_{\boldsymbol{D}_j}(\boldsymbol{x})]}^{\text{From } \hat{\mathtt{SC}}\left(\mathcal{A}, \{\hat{\boldsymbol{D}}_b\}_{b=1}^B, (\boldsymbol{x}, \boldsymbol{g})\right)} \Big] \\ &= \overbrace{\frac{2B_0B_1}{B(B-1)}}^{(C.6)} + \frac{1}{B(B-1)} \sum_{i \neq j} \mathbf{1}[h_{\boldsymbol{D}_i}(\boldsymbol{x}) = h_{\boldsymbol{D}_j}(\boldsymbol{x})]. \end{aligned}$$

Therefore, rearranging,

$$\hat{\mathtt{SC}}(\mathcal{A}, \hat{\mathbb{D}}, (\boldsymbol{x}, \boldsymbol{g})) = \frac{1}{B(B-1)} \sum_{i \neq j} \mathbf{1}[h_{\boldsymbol{D}_i}(\boldsymbol{x}) = h_{\boldsymbol{D}_j}(\boldsymbol{x})] = 1 - \frac{2B_0B_1}{B(B-1)}.$$

□

We note that $\hat{\mathtt{SC}}$ (3.2) is independent of specific costs C_{01} and C_{10}. Nevertheless, the choice of decision threshold τ will of course impact the values of B_0 and B_1 in practice. In turn, this will impact the degree of self-consistency

that a learning process exhibits empirically. In short, the measured degree of self-consistency in practice will depend on the choice of ℓ. Further, following an analysis similar to what we can show that $\hat{\mathbb{SC}}$ will be a value in $[0.5 + \epsilon, 1]$, for small positive ϵ. This reality is reflected in the results that we report for our experiments, for which $B = 101$ yields minimal $\hat{\mathbb{SC}} \approx 0.495$.

Cost-independence of self-consistency Intuitively, *self*-consistency of a learning process is a relative metric; it is a quantity that is measured relative to the learning process. We therefore conceive of it as a metric that is normalized with respect to the learning process (Definition 8). Such a process can be maximally 100% self-consistent, but it does not make sense for it to be more than that (reflected by the maximum value of 1).

In contrast, as discussed in Appendix C.2, variance can measure much greater than 1, depending on the magnitude of the sum of the costs C_{01} and C_{10}, in particular, for $C_{01} + C_{10} > 4$ (C.5). However, it is not necessarily meaningful to compare the magnitude of variance across classifiers. Recall that the effect of changing costs C_{01} and C_{10} corresponds to a change in the binary classification decision threshold, with $\tau = \frac{C_{01}}{C_{01}+C_{10}}$. It is the *relative* costs that change the decision threshold; not the costs themselves. For example, the classifier with costs $C_{01} = 1$ and $C_{10} = 3$ is equivalent to the classifier with costs $C_{01} = \frac{1}{2}$ and $C_{10} = \frac{3}{2}$ (for both, $\tau = \frac{1}{4}$), but the former would measure a larger magnitude for variance.

It is this observation that grounds our cost-independent definition of self-consistency in Section 3.3 and Appendix C.2.3. Given the fact that the magnitude of variance measurements can complicate our comparisons of classifiers,

as discussed above, we focus on the part of variance that encodes information about arbitrariness in a learning process: its measure of (dis)agreement between classification decisions that result from changing the training dataset. We could alternatively conceive of self-consistency as the additive inverse of normalized variance, but this is more complicated because it would require a computation that depends on the specific costs, $\hat{\texttt{var}}(\mathcal{A}, \hat{\mathbb{D}}, (\boldsymbol{x}, \boldsymbol{g}))_{\text{normalized}} = \frac{\hat{\texttt{var}}\left(\mathcal{A}, \hat{\mathbb{D}}, (\boldsymbol{x}, \boldsymbol{g})\right)}{\hat{\texttt{var}}\left(\mathcal{A}, \hat{\mathbb{D}}, (\boldsymbol{x}, \boldsymbol{g})\right)_{\text{max}}}$.

C.2.4 Additional details on our self-consistency metric

Terminology. In logic, the idea of consistent belief has to do with ensuring that we do not draw conclusions that contradcit each other. This is much like the case that we are modeling with self-consistency — the idea that underlying changes in the dataset can lead to predictions that are directly in contradition [277, 550, 560]. Ideas of consistency in legal rules have a similar flavor; legal rules should not contradict each other; legal judgments should not contradict each other (this is at least an aspiration for the law, based on common ideas in legal theory [218, 570]. For both of these reasons, the term "consistent" has a natural mapping to our usage of it in this paper. This is especially true in the legal theory case, given that inconsistency in the law is often considered arbitrary and a source of discrimination.

We nevertheless realize that the word "consistent" is overloaded with many meanings in statistics and different subfields computer science like distributed computing [3, 642, e.g.,]. Nevertheless, due to the clear relationship between our purposes concerning arbitrariness and discrimination, and definitions in logic and the law, we believe that it is the most appropriate term for our work.

Quantifying systematic arbitrariness. We depict *systematic arbitrariness* using the Wasserstein-1 distance [489]. This is the natural distance for us to consider because it has a closed form when being applied to CDFs. For our purposes, it should be interpreted as computing the total disparity in self-consistency by examining all possible self-consistency levels κ at once.

Formally,[1] for two groups $\boldsymbol{g} = 0$ and $\boldsymbol{g} = 1$ with respective $\texttt{SC}$ CDFs F_0 and F_1,

$$\mathcal{W}_1 = \int_{\mathbb{R}} |F_0(\kappa) - F_1(\kappa)| \, d\kappa.$$

For self-consistency, which we have defined on $[0.5, 1]$, this is just

$$\mathcal{W}_1 = \int_{0.5}^{1} |F_0(\kappa) - F_1(\kappa)| \, d\kappa.$$

Empirically, we can approximate this with

$$\hat{\mathcal{W}}_1 \coloneqq \frac{1}{|\hat{\mathbb{K}}|} \sum_{\hat{\mathbb{K}}} |\hat{F}_0(\hat{\kappa}) - \hat{F}_1(\hat{\kappa})|,$$
$$\text{where } \hat{\mathbb{K}} = \left\{ 1 - \frac{2B_0B_1}{B(B-1)} \middle| B_0 \in \{0 \dots B\} \wedge B_1 \in \{0 \dots B\} \wedge B_0 + B_1 = B \right\}.$$

[1] We consider the Wasserstein distance for one-dimensional distributions. More generally, the p-th Wasserstein distance for such distributions, $\mathcal{W}_p$, requires the inverse CDFs to be well-defined (i.e., the CDFs need to be strictly monotonic). This is fine to assume for our purposes. We have to relax the formal definition of the Wasserstein distance, anyway, when we estimate it in practice with a discrete number of samples.

We typically set $B = 101$, and thus

$$\hat{\mathbb{K}} = [0.49505, 0.49545, 0.49624, 0.49743, 0.49901, 0.50099, 0.50337, 0.50614, 0.50931, 0.51287, 0.51683, 0.52119, 0.52594, 0.53109, 0.53663, 0.54257, 0.54891, 0.55564, 0.56277, 0.57030, 0.57822, 0.58653, 0.59525, 0.60436, 0.61386, 0.62376, 0.63406, 0.64475, 0.65584, 0.66733, 0.67921, 0.69149, 0.70416, 0.71723, 0.73069, 0.74455, 0.75881, 0.77347, 0.78851, 0.80396, 0.81980, 0.83604, 0.85267, 0.86970, 0.88713, 0.90495, 0.92317, 0.94178, 0.96079, 0.9802, 1.0],$$

which we use to produce our CDF plots.

When measuring systematic arbitrariness with abstention, we set the probability mass for $< \kappa$ to 0 it. This makes sense because we are effectively re-defining the $\hat{\text{SC}}$ CDFs to not include instances that exhibit below a minimal amount of $\hat{\text{SC}}$. This also makes comparing systematic arbitrariness across CDFs for different interventions more interpretable. It allows us to keep the number of experimental samples for the empirical CDF measures constant when computing averages, so abstaining would then always have the effect of decreasing systematic arbitrariness. If we did not do this, because the Wasserstein-1 distance is an average, changing the set $\hat{\mathbb{K}}$, of course, would change the amount of Wasserstein-1 distance — possibly leading to a relative *increase* (if there are greater discrepancies between $\boldsymbol{g}$-condition CDF curves at $\geq \kappa$).

C.3 Related Work and Alternative Notions of Variance

As noted in Section 3.6, prior work that discusses variance and fair classification often relies on the definition of variance from Domingos [178]. We deviate

from prior work and provide our own definition for two reasons: 1) variance in Domingos [178, 179] does not cleanly extend to cost-sensitive loss, and 2) the reference point for measuring variance in Domingos [178, 179] — the *main prediction* — can be unstable/ brittle in practice. We start by explaining the Domginos [178, 179] definitions, and then use these definitions to support our rationale.

C.3.1 Defining variance in relation to a "main prediction"

To begin, we restate the definitions from Domingos [178, 179] concerning the expected model (called the *main predictor*). We change the notation from Domingos to align with our own, as we believe these changes provide greater clarity concerning meaning, significance, and consequent takeaways. Nevertheless, these definitions for quantifying error are equivalent to those in Domingos [179], and they fundamentally depend on human decisions for setting up the learning process.

Domgingos defines predictive variance in relation to this single point of reference. This reference point captures the general, expected behavior of models that could be produced by the chosen learning process. We can think of each prediction of this point of reference as the "central tendency" of the predictions made by all possible models in μ for $(\boldsymbol{x}, \boldsymbol{g})$. Formally,

Definition 9. *The **main prediction** $\hat{y}$ is the prediction value $y' \in \mathbb{Y}$ that generates the minimum average loss with respect to all of the predictions $\hat{y} \in \hat{\mathbb{Y}}$ generated by the different possible models in μ. It is defined as the expectation over training sets $\mathbb{D}$ for a*

loss function ℓ, given an example instance $(\boldsymbol{x}, \boldsymbol{g})$. That is,

$$\bar{y} = \arg\min_{y'} \mathbb{E}_{\mathbf{D}}[\ell(\hat{y}, y')|\mathbf{x} = \boldsymbol{x}, \mathbf{g} = \boldsymbol{g}]. \tag{C.7}$$

The main predictor $\bar{h} : \mathbb{X} \to \mathbb{Y}$ produces the main prediction $\bar{y}$ for each $(\boldsymbol{x}, \boldsymbol{g})$.

What (C.7) evaluates to in practice of course depends on the loss function ℓ. For squared loss, the main prediction is defined as the mean prediction of all the $h_{\boldsymbol{D}_k}$ [178, 325]. Following Kong and Dietterich [325], for 0-1 loss Domingos [178] defines the main prediction as the mode/majority vote — the most frequent prediction for an example instance $(\boldsymbol{x}, \boldsymbol{g})$. We provide a more formal discussion of why this is the case when we discuss problems with the main prediction for cost-sensitive loss (Appendix C.3.2). Domingos [178, 179] then define variance in relation to specific models $h_{\boldsymbol{D}_k}$ and the main predictor $\bar{h}$:

Definition 10. *The variance-induced error for fresh example instance $(\boldsymbol{x}, \boldsymbol{g})$ is*

$$\texttt{var}(\mathcal{A}, \mathbb{D}, (\boldsymbol{x}, \boldsymbol{g})) = \mathbb{E}_{\mathbf{D}}[\ell(\bar{y}, \hat{y})|\mathbf{x} = \boldsymbol{x}, \mathbf{g} = \boldsymbol{g}],$$

where $\bar{y} = \bar{h}(\boldsymbol{x})$ is the main prediction and the $\hat{y}$ are the predictions for the different $h_{\boldsymbol{D}_k} \sim \mu$.

That is, for a specific $(\boldsymbol{x}, \boldsymbol{g})$, it is possible to compare the individual predictions $\hat{y} = h_{\boldsymbol{D}_k}(\boldsymbol{x})$ to the main prediction $\bar{y} = \bar{h}(\boldsymbol{x})$. Using the main prediction as a reference point, one can compute the extent of disagreement of individual predictions with the main prediction as a source of error. It is this definition (Definition 10) that prior work on fair classification tends to reference when discussing variance [67, 119]. However, as we discuss in more detail below (Appendix C.3.2), many of the theoretical results in Chen et al. [119] follow directly from the definitions in Domingos [178], and the experiments do not actually use

those results in practice. Black et al. [67], in contrast, presents results that rely heavily on the main prediction in Domingos [178].

C.3.2 Why we choose to avoid computing the main prediction

We now compare our definition of variance (Definition 9) to the one in Domingos [178, 179] (Definition 10). This comparison makes clear in detail why we deviate from prior work that relies on Domingos [178, 179].

No decomposition result. Following from above, it is worth noting that by not relying on the main prediction, we lose the applicability of the decomposition result that Domingos [178, 179] develops. However, we believe that this is fine for our purposes, as we are interested in the impact of empirical variance specifically on fair classification outcomes. We do not need to reason about bias or noise in our results to understand the arbitrariness with which we are concerned (Section 3.3.1). It is also worth noting that prior work on fair classification that leverages Domingos [178] also does not leverage the decomposition, either. Chen et al. [119] extends the decomposition to subgroups in the context of algorithmic fairness,[2] and then informally translates the takeaways of the Domingos [178] result to a notion of a "level of discrimination." Moreover, unlike our work, these prior studies do not actually measure variance directly in its experiments.

[2]This just involves splitting the conditioning on an example instance of features $\boldsymbol{x}$ into conditioning on an example instance whose features are split into $(\boldsymbol{x}, \boldsymbol{g})$.

No need to compute a "central tendency." In Domingos [178, 179], variance is defined in terms of both the loss function ℓ and the main prediction $\bar{y}$. This assumes that the main prediction is well-defined for the loss function, and that it is well-behaved. While there is a simple interpretation of the main prediction for squared loss (the mean) and for 0-1 loss (the mode/majority vote), it is significantly messier for cost-sensitive loss, which is a more general formulation that includes 0-1 loss. Domingos [178, 179] does not discuss this explicitly, so we derive the main prediction for cost-sensitive loss ourselves below. In summary:

- The behavior of the main prediction for cost-sensitive loss reveals that the decomposition result provided in the extended technical report (Theorem 4, Domingos [179]) is in fact very carefully constructed. We believe that this construction is so specific that it is not practically useful (it is, in our opinion, hardly "unified" in a more general sense, as it is so carefully adapted to specific loss functions and their behavioral special cases).
- By decoupling from the need to compute a main prediction as a reference point, our variance definition is ultimately much simpler and more general, with respect to how it accommodates different loss functions.[3]

Brittleness of the main prediction. For high variance instances, the main prediction can flip-flop from $\hat{y} = 1$ to $\hat{y} = 0$ and back. While the strategy in Black et al. [67] is to abstain on the prediction in these cases, we believe that a bet-

[3]This reveals a subtle ambiguity in the definition of the loss ℓ in Domingos [178, 179]. Neither paper explicitly defines the signature of ℓ. For the main prediction (Definition 9) and variance (Definition 10), there is a lack of clarity in what constitutes a valid domain for ℓ. Computing the main prediction $\bar{y}$ suggests $\ell : \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}_{\geq 0}$, where $\bar{y} \in \mathbb{Y}$, but, since $\hat{\mathbb{Y}} \subseteq \mathbb{Y}$, it is possible that $\bar{y} \notin \mathbb{Y}$. However, the definition of variance suggests that $\ell : \mathbb{Y} \times \hat{\mathbb{Y}} \to \mathbb{R}_{\geq 0}$. Since $\hat{\mathbb{Y}} \subseteq \mathbb{Y}$, it is not guaranteed that $\hat{\mathbb{Y}} = \mathbb{Y}$. This may be fine in practice, especially for squared loss and 0-1 loss (the losses with which Domingos [178] explicitly contends), but it does arguably present a problem formally with respect to generalizing.

ter alternative is to understand that the main prediction is not very meaningful more generally for high-variance examples. That is, for these examples, the ability (and reliability) of breaking close ties to determine the main (simple majority) prediction is not the right approach. Instead, we should ideally be able to embed more confidence into our process than a simple-majority-vote determination.[4] Put different, in cases for which we can reliably estimate the main prediction, but the vote margin is slim, we believe that the main prediction is still uncertain, based on our understanding of variance, intuited in Figure 3.1. **The main prediction can be reliable, but it can still, in this view, be arbitrary** (Section 3.6). With a simple-majority voting scheme, there can be huge differences between predictions that are mostly in agreement, and those that are just over the majority reference point. Freeing ourselves of this reference point via our self-consistency metric, we can define thresholds of self-consistency as our criterion for abstention (where simple-majority voting is one instantiation of that criterion).[5]

The main prediction and cost-sensitive loss

We show here that, for cost-sensitive loss, the main prediction depends on the majority class being predicted, the asymmetry of the costs, and occasional tie-breaking, such that the main prediction can either be the majority vote or the minority vote. Domingos [179] provides an error decomposition in Theorem 4,

[4]This is also another aspect of the simplicity of not needing to define and compute a "central tendency" prediction. We do not need to encode a notion of a tie-breaking vote to determine a "central tendency." The main prediction can be unclear in cases for which there is no "main outcome" (e.g., Individual 2 in Figure 3.1), as the vote is split exactly down the middle. By avoiding the need to vote on a main reference point, we also avoid having to ever choose that reference point arbitrarily.

[5]This problem is worse for cost-sensitive loss, where the main prediction is not always the majority vote (see below).

but does not explain the effects on the main prediction. We do so below, and also call attention to 0-1 loss as a special case of cost-sensitive loss, for which the costs are symmetric (and equal to 1). We first summarize the takeaways of the analysis below:

- **Symmetric loss**: The main prediction is the **majority vote**.
- **Asymmetric loss**: Compute 1) the relative cost difference (i.e., $\frac{C_{01}-C_{10}}{C_{10}}$), 2) the majority class (and, as a result, the minority class) for the $\hat{y} \in \hat{\mathbb{Y}}$, and 3) the relative difference in the number of votes in the majority and minority classes (i.e., what we call the *vote margin*; below, $\frac{(i+2j+1)-i}{i}$)
  - If the **majority class** in $\hat{\mathbb{Y}}$ has the **lower cost** of misclassification, then the main prediction is the **majority vote**.
  - If the **majority class** in $\hat{\mathbb{Y}}$ has the **higher cost** of misclassification, then the main prediction **depends on the asymmetry of the costs and the vote margin**, i.e.,
    * If $\frac{C_{01}-C_{10}}{C_{10}} = \frac{(i+2j+1)-i}{i}$, we can choose the main prediction to be **either class** (but must make this choice consistently).
    * If $\frac{C_{01}-C_{10}}{C_{10}} > \frac{(i+2j+1)-i}{i}$, the **minority vote** is the main prediction.
    * If $\frac{C_{01}-C_{10}}{C_{10}} < \frac{(i+2j+1)-i}{i}$, the **majority vote** is the main prediction.

Proof. Let us consider cost-sensitive loss for binary classification, for which $\ell(0,0) = \ell(1,1) = 0$ and we have potentially-asymmetric loss for misclassifications, i.e. $\ell(1,0) = C_{10}$ and $\ell(0,1) = C_{01}$, with $C_{01}, C_{10} \in \mathbb{R}^+$. 0-1 loss is a special case for this type of loss, for which $C_{01} = C_{10} = 1$.

Let us say that the total number of models trained is k, which we evaluate on an example instance $\boldsymbol{x}$. Let us set $|\hat{\mathbb{Y}}| = k = 2i + 2j + 1$, with $i \geq 0$ and $j \geq 0$. We

can think of i as the common number of votes that each class has, and $2j + 1$ as the margin of votes between the two classes. Given this setup, this means that $k \geq 1$, i.e., we always have the predictions of at least 1 model to consider, and k is always odd. This means that there is always a strict majority classification.

Without loss of generality, on $\boldsymbol{x}$, of these k model predictions $\hat{y} \in \hat{\mathbb{Y}}$, there are i class-0 predictions and $i + 2j + 1$ class-1 predictions (i.e., we do our analysis with class 1 as the majority prediction). To compute the main prediction $\bar{y}$, each $\hat{y} \in \hat{\mathbb{Y}}$ will get compared to the values of possible predictions $y' \in \mathbb{Y} = \{0, 1\}$. That is, there are two cases to consider:

- **Case** $y' = 0$: $y' = 0$ will get compared i times to the i $\hat{y} = 0$s in $\hat{\mathbb{Y}}$, for which $\ell(0, 0) = 0$; $y' = 0$ will similarly get compared $i + 2j + 1$ times to the 1s in in $\hat{\mathbb{Y}}$, for which (by Definition 9) the comparison is $\ell(1, 0) = C_{10}$. By definition of expectation, the expected loss is

$$\frac{i \times 0 + (i + 2j + 1) \times C_{10}}{2i + 2j + 1} = \frac{C_{10}(i + 2j + 1)}{2i + 2j + 1}. \tag{C.8}$$

- **Case** $y' = 1$: Similarly, the label 1 will also get compared i times to the 0s in $\hat{\mathbb{Y}}$, for which the comparison is $\ell(0, 1) = C_{01}$; $y' = 1$ will also be compared $i + 2j + 1$ times to the 1s in $\hat{\mathbb{Y}}$, for which $\ell(1, 1) = 0$. The expected loss is

$$\frac{i \times C_{01} + (i + 2j + 1) \times 0}{2i + 2j + 1} = \frac{C_{01} i}{2i + 2j + 1}. \tag{C.9}$$

We need to compare these two cases for different possible values of C_{10} and C_{01} to understand which expected loss is minimal, which will determine the main prediction $\bar{y}$ that satisfies Equation (C.7). The three different possible relationships for values of C_{10} and C_{01} are $C_{10} = C_{01}$ (symmetric loss), and $C_{10} > C_{01}$ and $C_{10} < C_{01}$ (asymmetric loss). Since the results of the two cases above share

the same denominator, we just need to compare their numerators, $C_{10}(i + 2j + 1)$ (C.8) and $C_{01}i$ (C.9).

Symmetric Loss (0-1 Loss). When $C_{10} = C_{01} = 1$, the numerators in (C.8) and (C.9) yield expected losses $i + 2j + 1$ and i, respectively. We can rewrite the numerator for (C.9) as

$$i + \overbrace{2j + 1}^{\geq 1, \text{ given } j \geq 0} \geq i + 1,$$

which makes the comparison of numerators $i < i + 1$, i.e., we are in the case (C.9) < (C.8). This means that the case of $y' = 1$ (C.9) is the minimal one; the expected loss for class 1, the most frequent class, is the minimum, and thus the most frequent/ majority vote class is the main prediction. An analogous result holds if we instead set the most frequent class to be 0. More generally, this holds for all symmetric losses, for which $C_{10} = C_{01}$.

► For **symmetric losses**, the main prediction $\bar{y}$ is **majority vote** of the predictions in $\hat{\mathbb{Y}}$.

Asymmetric Loss. For asymmetric/ cost-sensitive loss, we need to examine two sub-cases: $C_{10} > C_{01}$ and $C_{10} < C_{01}$.

- **Case $C_{10} > C_{01}$**: $C_{01}i < C_{10}(i + \overbrace{2j + 1}^{\geq 1})$, given that $j \geq 0$. Therefore, since $C_{01}i$ is minimal and associated with class 1 (the most frequent class in our setup), the majority vote is the main prediction. We can achieve an analogous result if we instead set 0 as the majority class.

  ► For **asymmetric losses**, the main prediction $\bar{y}$ is the **majority vote** of

the predictions in $\hat{\mathbb{Y}}$, **if the majority class has a cheaper cost associated with misclassification** (i.e., if the majority class is 1 and $C_{10} < C_{01}$, or if the majority class is 0 and $C_{01} < C_{10}$).

- **Case $C_{10} < C_{01}$**: If $C_{10} < C_{10}$, it depends on how asymmetric the costs are and how large the vote margin (i.e., $2j + 1$) between class votes is. There are 3 sub-cases:

  - **Case $C_{01}i = C_{10}(i + 2j + 1)$, i.e. cost equality**: We can look at the relative asymmetric cost difference of the minority class cost (above C_{01}, without loss of generality) and the majority class cost (above C_{10}, without loss of generality), (above $\frac{C_{01}-C_{10}}{C_{10}}$, without loss of generality). If that relative cost difference is equal to the relative difference of the votes between the majority and minority classes (i.e., $\frac{(i+2j+1)-i}{i}$), then the costs of predicting either 1 or 0 are equal. That is, we can rearrange terms as a ratio of costs to votes:

$$
\begin{aligned}
C_{01}i &= C_{10}(i + \overbrace{2j + 1}^{\geq 1}) && \text{(The terms in this equality are } > 0) \\
\frac{C_{01}}{C_{10}} &= \frac{i + 2j + 1}{i} && \text{(Given the above, } C_{01}i > 0 \text{ so } i > 0) \\
&= 1 + \frac{2j + 1}{i} \\
\frac{C_{01}}{C_{10}} - 1 &= \frac{2j + 1}{i} \\
\frac{C_{01} - C_{10}}{C_{10}} &= \frac{2j + 1}{i} = \frac{(i + 2j + 1) - i}{i} \geq \frac{1}{i} && \text{(C.10)}
\end{aligned}
$$

    ▶ For asymmetric loss **when the majority-class-associated cost is less than the minority-class associated cost and if the expected losses are equal**, then the **main prediction $\bar{y}$ is either** 1 **or** 0, (and we must make this choice consistently).

– **Case** $C_{01}i > C_{10}(i + 2j + 1)$: We can look at the relative asymmetric cost difference of the minority class cost (above C_{01}, without loss of generality) and the majority class cost (above C_{10}, without loss of generality), (above $\frac{C_{01}-C_{10}}{C_{10}}$, without loss of generality). If that relative cost difference is greater than the relative difference of the votes between the majority and minority classes (i.e., $\frac{(i+2j+1)-i}{i}$), then the *minority vote* yields the minimum cost and is the main prediction $\bar{y}$ (above $\bar{y} = 0$, without loss of generality; an analogous result holds if we had set the majority vote to be 0 and the minority vote to be 1). Following (C.10) above, this is the same as

$$\frac{C_{01} - C_{10}}{C_{10}} > \frac{(i + 2j + 1) - i}{i}$$

► For asymmetric loss **when the majority-class-associated cost is less than the minority-class associated cost**, it is possible for the **minority class** to have a greater associated loss. In this case, the ***minority vote*** **is the main prediction** $\bar{y}$.

– **Case** $C_{01}i < C_{10}(i + 2j + 1)$: We can look at the relative asymmetric cost difference of the minority class cost (above C_{01}, without loss of generality) and the majority class cost (above C_{10}, without loss of generality), (above $\frac{C_{01}-C_{10}}{C_{10}}$, without loss of generality). If that relative cost difference s less than the relative difference of the votes between the majority and minority classes (i.e., $\frac{(i+2j+1)-i}{i}$), then the majority vote yields to minimum cost and is the main prediction $\bar{y}$ (above $\bar{y} = 1$, without loss of generality; an analogous result holds if we had set the majority vote to be 0 and the minority vote to be 1). Following (C.10) above, this is the same as

$$\frac{C_{01} - C_{10}}{C_{10}} < \frac{(i + 2j + 1) - i}{i}$$

▶ For asymmetric loss **when the majority-class-associated cost is less than the minority-class associated cost**, it is possible for the **majority class** to have a greater associated loss. In this case, the ***majority vote* is the main prediction** $\bar{y}$.

□

C.3.3 Putting our work in conversation with research on model multiplicity

A line of related work to ours concerns *model multiplicity* and fairness [68, 395, 616]. This work builds off of an observation made by Breiman [86] regarding how there are multiple possible models of the same problem that exhibit similar degrees of accuracy. This set of multiple possible models of similar accuracy is referred to as the Rashomon set [86].

Work on model multiplicity has recently become fashionable in algorithmic fairness. In an effort to develop more nuanced model selection metrics beyond looking at just fairness and accuracy for different demographic groups, work at the intersection of model multiplicity and fairness tends to examine other properties of models in the Rashomon set in order to surface additional metrics for determining which model to use in practice.

At first glance, this work may seem similar to what we investigate here, but

we observe four key differences:[6]

1. Model multiplicity places conditions on accuracy and fairness in order to determine the Rashomon set. We place no such conditions on the models that a learning process (Definition 8) produces; we simulate the distribution over possible models μ without making any claims about the associated properties of those models.

2. Model multiplicity makes observations about the Rashomon set with the aim of still ultimately putting forward criteria for helping to select *a single model*. While the metrics used to inform these criteria include variance, most often work on model multiplicity still aims to choose one model to use in practice.

3. Much of the work on model multiplicity emphasizes theoretical contributions, whereas our emphasis is on more experimental contributions. In conjunction with the first point, of ultimately trying to arrive at a single model, this work is also trying to make claims with respect to the Bayes-optimal model. Given our empirical focus — of what we can actually produce in practice — claims about optimality are not our concern.

4. We focus specifically on variance reduction as a way to mitigate arbitrariness. We rely on other work, coincidentally contributions also made by Breiman, to study arbitrariness [84], and emphasize the importance of using ensemble models to produce predictions or abstention from prediction. We do not study the development of model selection criteria to pick a single model to use in practice; we use self-consistency to give a sense of predictive confidence about when to predict or not. We always select

[6]We defer discussion of Black et al. [67] to C.3.4.

an ensemble model — regardless of whether that model is produced by simple or super ensembling (Section 3.4) — and then use a user-specified level of self-consistency κ to determine when that model actually produces predictions.

These differences ultimately lead to very different methods for making observations about fairness. Importantly, we can study the arbitrariness of the underlying laerning process with a bit more nuance. For example, it could be the case that a particular task is just impossible to get right for some large subset of the test data (and this would be reflected in the Rashomon set of models), but for some portion of it there is a high amount of self-consistency for which we may still want to produce predictions.

Further, based on our experimental approach, we highlight completely different normative problems than those highlighted in work on model multiplicity (notably, see Black et al. [68]). So, in short, while model multiplicity deals with related themes as our work — issues of model selection, problem formulation, variance, etc. — the goals of that work are ultimately different, but potentially complementary, from those in our paper.

For example, a potentially interesting direction for future work would be to measure how metrics from work on model multiplicity behave in practice in light of the ensembling methods we present here. We could run experiments using Algorithm 2 and investigate model multiplicity metrics for the underlying ensembled models. However, we ultimately do not see a huge advantage to doing this. Our empirical results indicate that variance is generally high, and has led to reliability issues regarding conclusions about fairness and accuracy. In fairness settings and available benchmarks, we find that the most important

point is that variance has muddled conclusions. Under these circumstances, ensembling with abstention based on self-consistency seems a reasonable solution, in contrast to finding a single best model in the Rashomon set that attains other desired criteria.

C.3.4 Concurrent work

There are several related papers that either preceded or came after this work's public posting. Some of this work is clearly concurrent, given the time frame. Other works that came after ours are not necessarily concurrent, but are either independent and unaware of our paper, or build on our work.

Setting the stage in 2021. The present work was scoped in 2021, in direct response to the initial study by Forde et al. [210] and critical review by Cooper and Abrams [136]. Forde et al. [210] was one of the first (if not the first) paper to note that variance is overlooked in problem formulations that consider fairness. However, it was limited in scope and also dealt with deep learning settings, which have multiple sources of non-determinism that can be difficult to tease apart with respect to their effects on variance.

Cooper and Abrams [136] notes important, overlooked normative assumptions in the fairness-accuracy trade-off problem formulation, and suggests that this formulations is tautological. Our work is a natural direction for future research, in this respect – to see how, in practice, the fairness-accuracy trade-off behaves after we account for variance. Indeed, we find that there is often no such trade-off, but for different reasons than those suggested by Cooper and

Abrams [136]. We expected there to be residual label bias that contributes to noise-induced error, but ultimately did not really observe this in practice. In these respects, our work both strengthens and complements these prior works. We support their claims, and go significantly beyond the work they did in order to provide such support. Further, our results suggest additional conclusions about experimental reliability in algorithmic fairness.

Variance and abstention-based ensembling. Black et al. [67] is concurrent work that slightly preceded our public posting. This work is similarly is interested in variance reduction, ensembling, and abstention in fairness settings, but fundamentally studies these topics in a different manner. We address four differences:

1. Black et al. [67] does not take the wide-ranging experimental approach that we take. While we both study variance and fairness, our work also considers *the practice of fair classification research* as an object of study. It is for these reasons that we do so many experiments on benchmark datasets, and clean and release another dataset for others to use.
2. They rely on the definition of variance from Domingos [178] in their work, likely building on the choice made by Chen et al. [119] to use this defintion. Much of this Appendix is devoted to discussing Domingos [178, 179] and his definition of variance. The overarching takeaway from our discussion is that 1) there are technical problems with this definition (which have been noted by others that investigated the bias-variance-noise trade-off for 0-1 loss in the early 2000s), 2) the definition does not naturally extend to cost-sensitive loss, 3) the main prediction can be unstable in practice and

thus should not be the criterion for investigating arbitrariness (indeed, relying on the main prediction just pushes arbitrariness into that definition). While Black et al. [67] observes that variance is an important consideration for fairness, they ultimately focus on reliable estimation of the main prediction as the criterion for abstention in their ensembling method. While this kind of reliability is important, it does not deal with the general problem of arbitrary predictions (i.e., it is possible to have a reliable main prediction that is still effectively arbitrary). As a result, the nature of when and how to abstain is very different from ours. We instead base our criterion on a notion of confidence in the prediction, and we allow for flexibility around when to abstain when predictions are too arbitrary.

3. As a result of the above two differences, the claims and conclusions in both of our works are different. While there are similar terms used in both works (e.g., variance, abstention), which may make the works seem overlapping with a cursory read, our definitions, methods, claims, and conclusions are non-overlapping. For example, as stated in 1., while Black et al. [67]'s use of successful ensembles is intended to address individual-level arbitrariness, by relying on traditional bagging (simple-majority vote ensembling) and the definition of variance from Domingos [178] that encodes a main prediction, arbitrariness gets pushed into the aggregation rule. If they can estimate the mode prediction reliably, they do not abstain; the mode, however, may still be effectively arbitrary. Our measure of arbitrariness is more direct and more configurable. We can avoid such degenerate situations, as in the example we give for making reliable but arbitrary predictions in Black et al. [67].

4. We also describe a method for recursively ensembling in order to achieve

different trade-offs between abstention and prediction. This type of strategy is absent from Black et al. [67].

Deep learning. Qian et al. [477] is work that came after Forde et al. [210]. They, too, do a wide-ranging empirical study of variance and fairness, but focus on deep learning settings. As a result, they are not examining the fair classification experimental setup that is most common in the field. They therefore make different claims about reliability, which have a similar flavor as those that we make here. However, because of our setup, we are able to probe these claims much deeper (due in part to model/ problem size and being able to limit nondeterminism solely to sampling the training data). We mention this work because of its close relationship to Forde et al. [210], which in part inspired this study.

Ko et al. [320] is another deep learning fairness paper. It was posted publicly months after our study, and examines non-overlapping settings and tasks. While the results are similar — we find fairness after ensembling — it is again fundamentally different (along the lines of Qian et al. [477] and Forde et al. [210]) because it does not study common non-deep-learning setups. They also do not study arbitrariness, which is one of the main purposes of our paper.

Variance in fair classification. Khan et al. [312] is concurrent work that studies the same problem that we study, but also takes a different approach. For one, they bake in a notion of 0-1 loss into their definitions. In this respect, our definition of self-consistency generalizes the definitions in their paper. While they run more types of models than we do (we initially ran more, but ultimately stopped because the results were largely similar with more common model types), they

do not cover as many datasets as we do. They also do not study arbitrariness or abstention-based ensembling to deal with it, and they do not release a dataset. Further, based on the fact that they study fewer empirical tasks than we do, and that they do not examine abstention-based ensembling, they do not surface or make claims about the experimental reliability issues that we observe. They do not make claims about the fundamental problem that we observe: **That variance is the culprit for much observed algorithmic unfairness in classification; in practice, we do not seem to learn very confident decisions for large portions of the datasets we examine, and this is a key problem that has been masked by current common experimental practices in the field**.

Other work. Any other work on variance and fairness **comes after** the present study. We have made a significant attempt to keep our related work section up-to-date in response to this new work. We have used a detailed and robust mixed of Google alerts and scraping arXiv to find new related work. We used this same procedure to make sure we found (ideally) all related work on fairness and variance when we conducted this project. There are some studies, which directly build on ours, which we choose not to cite.

C.4 Additional Details on Our Algorithmic Framework

A natural question is to see if we can improve self-consistency, with the hope that doing so would reduce arbitrariness in the learning process, improve accuracy, and, for the cases in which there is different self-consistency across subgroups, also perhaps improve fairness. To do so, we consider ways of reducing

variance, as, based on our definitions (Definition 9 and 10), doing so should improve self-consistency.

We consider the classic *b*ootstrap *agg*regation — or, *bagging* — algorithm [84] as a starting point. It has been well-known since Breiman [84] that *bagging* can improve the performance of unstable predictors. That is, for models produced by a learning process that is sensitive to the underlying training data, it is (theoretically-grounded) good practice to train an ensemble of models using bootstrapping (Appendix C.1.4; Efron [184]; Efron and Tibshirani [186]). When classifying an example instance, we then leverage the whole ensemble by aggregating the predictions produced by its members. This aggregation process identifies the most common prediction in the ensemble, and returns that label as the classification. Put differently, we have combined the information of a lot of unstable classifiers, and averaged over their behavior in order to generate more stable classifications.

Given the the relationship between variance (Definition 9) and self-consistency (Definition 10), reducing variance will improve self-consistency. However, rather than relying on a simple-majority-vote to decide the aggregated prediction, we also will instill a notion of confidence in our predictions by requiring a minimum level of self-consistency, which is described in Algorithm 2.

C.4.1 Self-consistent ensembling with abstention

We present a framework that alters the semantics of classification outputs to 0, 1, and `Abstain`, and employ ensembling to determine the $\widehat{\texttt{SC}}$-level that guides

the output process. We modify bagging from using a simple-majority-vote because this type of aggregation rule still allows for arbitrariness. If, for example, we happen to train $B = 101$ classifiers, it is possible that 50 of them yield one classification and the other 51 yield the other classification for a particular example. Bagging would select the classification that goes along with the 51 underlying models; however, if we happened to train $B = 103$ models, it is perhaps the case that the majority vote would flip. In short, the bagging aggregation rule bakes in the idea that simple-majority voting is a sufficient strategy for making decisions. And while this may generally be true for variance reduction in high-variance classifiers, it does not address the problem of arbitrariness that we study. It just encodes arbitrariness in the aggregation rule — it picks classifications, in some cases, that are no better than a coin flip.

Instead, Algorithm 2 is more flexible. It suggests many possible ways to produce bagged classifiers that do not have to rely on simple-majority voting, by allowing for abstentions. For example, we can change the aggregation rule in regular bagging to use a self-consistency level κ rather than majority vote. Instead of relying on votes, we can bag the underlying prediction probabilities and then apply κ a filter. We could take the top-n most consistent predictions and let a super-ensemble of underlying bagged classifiers decide whether to abstain or predict.

In the experiments in the paper, we provide two examples: Changing the underlying bagging vote aggregation rule (simple ensembling), and applying a round of regular bagging to do variance reduction and then bagging the bagged outputs (super ensembling) to apply a self-consistency threshold. Our ensemble model will not produce predictions for examples for which the lack of self-

consistency is too high. We describe our procedure more formally in Algorithm 2.

Simple proof that abstention improves self-consistency (by construction). We briefly show the simple proof that any method that meets the semantics of Algorithm 2 will be more self-consistent than its counterpart that cannot `Abstain`.

We define abstentions to be in agreement with both 0 and 1 predictions. This makes sense intuitively: Algorithm 2 abstains to avoid making predictions that lack self-consistency, so abstaining should not increase disagreement between predictions.

It follows that we can continue to use Definition 10 and associated empirical approximations $\hat{\mathtt{SC}}$ (3.3), but with one small adjustment. Instead of the total number of predictions $B = B_0 + B_1$, with B_0 and B_1 corresponding to 0 and 1 predictions, respectively, we now allow for $B \geq B_0 + B_1$, in order to account for possibly some non-zero number of abstentions.

In more detail, let us denote $\hat{\mathbb{Y}}$ to be the multiset of predictions for models $h_{\boldsymbol{D}_1}, h_{\boldsymbol{D}_2}, \ldots, h_{\boldsymbol{D}_B}$ on $(\boldsymbol{x}, \boldsymbol{g})$, with $|\hat{\mathbb{Y}}| = B = B_0 + B_1 + B_{\mathtt{Abstain}}$. This is where we depart from our typical definition of self-consistency, for which $B = B_0 + B_1$ (Section 3.3, Appendix C.2.3). We continue to let B_0 and B_1 represent the counts of 0 and 1 predictions, respectively, and now include $B_{\mathtt{Abstain}}$ to denote the (possibly nonzero) number of abstentions. This leads to the following adjustment of (3.3):

$$\hat{\mathtt{SC}}(\mathcal{A}, \{\hat{\boldsymbol{D}}_b\}_{b=1}^{B}, (\boldsymbol{x}, \boldsymbol{g})) = 1 - \frac{2(B_0 B_1 + B_0 B_{\mathtt{Abstain}} + B_1 B_{\mathtt{Abstain}})}{B(B-1)}. \tag{C.11}$$

Equation (C.11) follows from a similar analysis of comparing 0s, 1s, and absten-

tions for Definition 10, which lead us to derive (3.3) in Appendix C.2.3. However, since the costs of 0-to-`Abstain` comparisons and 1-to-`Abstain` comparisons are both 0, the $B_0 B_{\texttt{Abstain}}$ and $B_1 B_{\texttt{Abstain}}$ terms in (C.11) reduce to 0. As a result, we yield our original definition for self-consistency (3.3), with the possibility that $B = B_0 + B_1 + B_{\texttt{Abstain}} > B_0 + B_1$, if there is a nonzero number of abstentions $B_{\texttt{Abstain}}$.

Since $B > 1$ and $B_0, B_1, B_{\texttt{Abstain}} \geq 0$, it is always the case that option to `Abstain` is at least as self-consistent as not having the option to do so. This follows from the fact that $B_0 + B_1 + B_{\texttt{Abstain}} = B \geq B_0 + B_1$, which would make the denominator in (C.11) greater than or equal to the corresponding method that cannot `Abstain`; when subtracted from 1, this would produce a $\hat{\texttt{SC}}$ that is no smaller than the value for the corresponding method without that cannot `Abstain`.

Now, it follows that, given the choice between `Abstain` and predicting a label that is in disagreement with an existing prediction label, choosing to `Abstain` will always lead to higher self-consistency. This is because the cost to `Abstain` is less than disagreeing, so it will always be the minimal choice that maximizes $\hat{\texttt{SC}}$.

Error and the abstention set. It is very straightforward to see that the *abstention set* will generally exhibit higher than the *prediction set*. When we ensemble and measure $\hat{\texttt{SC}}$, the exmaples that exhibit low $\hat{\texttt{SC}}$ contain higher variance-induced error. Let us call the size of the abstention set U (which incurs error u), the size of the prediction set V (which incurs error v), and the size of the test set T (which incurs error t). We can relate the total number of misclassified examples

as $T * t = U * u + V * v$, with $T = U + V$. If we assume the bias and noise are equally distributed across the test and abstention sets (this is a reasonable assumption, on average, in our setup), then splitting off the high variance instances from the low variance (high $\widehat{\texttt{SC}}$ instances) requires that $u > v$. The error on the abstention set necessarily has to be larger than the error on the prediction set, in order to retain the above relationship.

C.5 Additional Experimental Results and Details for Reproducibility

The code for the examples in Sections 3.1, 3.3 and 3.5 can be found in through our paper on arXiv. This repository also contains necessary and sufficient information concerning reproducibility. At the time of writing, we use `Conda` to produce environments with associated package-versioning information, so that our results can be exactly replicated and independently verified. We also use the `Scikit-Learn` [465] toolkit for modeling and optimization. More details on our choice of models and hyperparameter optimization can be found in our code repository, cited above. In brief, we consulted prior related work (e.g., Chen et al. [119]) and performed our own validation for reasonable hyperparameters per model type. We keep these settings fixed to reduce impact on our results, in order to observe in isolation how different training data subsets impact our results.

During these early runs, we collected information on train accuracy, not just test accuracy; while models ultimately have similar test accuracy in most cases for the same task, they can vary significantly in terms of train accuracy (e.g., for

logistic regression, COMPAS is in the low .70s; for random forests, it is in the mid .90s). We do not include these results for the sake of space.

This section is organized as follows. We first present information on our datasets, models and code, including our HDMA toolkit (Appendix C.5.1). We then provide details on our setup for running experiments on our cluster (Appendix C.5.2). Appendix C.5.3 contains more detailed information concerning the experiments performed to produce Figures 3.1 and 3.2 in the main paper. In Appendix C.5.4, we discuss implications of these results for common fairness benchmarks like South German Credit. We defer more extensive results on our ensembling algorithm to our online appendix.

Note on CDF figures. We show our results in terms of the $\widehat{\text{SC}}$ of the underlying bagged models because doing so conveys how Algorithm 2 makes decisions to predict or abstain.[7] For both types of ensembling, Algorithm 2 predicts for all examples captured by the area to the right of the κ reference line, and abstains for all examples on the left.

A remark on cost It can be considerably more computationally intensive to train an ensemble of models to compute $\widehat{\text{SC}}$ than to train a handful of models and perform cross-validation, as is the standard practice in fair classification. However, as our empirical analysis demonstrates, this cost comes with a huge benefit: It enables us to improve self-consistency and to root out the arbitrariness of producing predictions that are effectively close-to-random, which is especially important in high-stakes fairness settings [144]. Moreover, for common

[7] The $\widehat{\text{SC}}$ CDF of Algorithm 2, computed via a *third* round of bootstrapping, has nearly all mass at $\widehat{\text{SC}} = 1$; it is difficult to visualize.

fair classification datasets, the increased cost on modern hardware is relatively small; (super-) ensembling with confidence takes under an hour to execute (online appendix).

C.5.1 Hypothesis classes, datasets, and code

Models. According to a comprehensive recent survey study [194], as well as related work like Chen et al. [119], we conclude that some of the most common models used in fair classification are logistic regression, decision tree classifiers, random forest classifiers, SVMs, and MLPs. We opted to include comprehensive results for the first three, since they capture different complexities, and therefore encode different degrees of statistical bias, that we expected to have an impact on the underlying sources of error. Since we choose not to use stochastic optimizers to reduce the sources of randomness, for our results, training MLPs is slower than it could be. We consistently use a decision threshold of 0.5 (i.e., 0-1 loss) for our experiments, though our results can easily be extended to other thresholds, as discussed in Section 3.3. Depending on the dataset, we reserve between 20% and 30% of the available data for the test set. This is consistent with standard fair classification training settings, which we validated during our initial experiments to explore the space (for which we also did preliminary hyperparameter optimization, before fixing the hyperparameters for our presented results).[8]

Datasets. Also according to Fabris et al. [194], the most common tasks in fair classification are `Old Adult` [322], `COMPAS` [344], and `South German`

[8]Please refer to the arXiv paper version for more details.

`Credit` [248].[9] These three datasets arguably serve as a *de facto* benchmark in the community, so we felt the need to include them in the present work. In recognition of the fact that these three datasets, however standard, have problems, we also run experiments on 3 tasks in the `New Adult` dataset, introduced by Ding et al. [174] to replace `Old Adult`. We subset to the `CA` (California) subset of the dataset, and run on `Income`, `Employment`, and `Public Coverage`, and consider `sex` and `race` as protected attributes, which we binarize into {Male, Female} and {White, Non-white}. These are all large-scale tasks, at least in the domain of algorithmic fairness — on the order of hundreds of thousands of example instances. However, the 3 tasks do share example instances and some features. In summary, concerning common tasks in fair classification:

- `COMPAS` [344]. We run on the commonly-used version of this dataset from Friedler et al. [213], which has 6167 example instances with 404 features. The target is to predict recidivism within 2 years (1 corresponding to Yes, and 0 to No). The protected attribute is `race`, binarized into "Non-white" (0) and "White" (1) subgroups.
- `Old Adult` [322]. We run on the commonly-used version of this dataset from Friedler et al. [213], which has 30,162 examples with 97 features. This version of the dataset removes instances with missing values from the original dataset, and changes the encoding of some of the features (Kohavi [322] has 48842 example instances with 88 features). The target is to predict $< \$50,000$ income (0) $>= \$50,000$ income (1). The protected attribute is `sex`, binarized into "Female" (0) and "Male" (1) subgroups.
- `South German Credit` [248]. We download the dataset from UCI[10]

[9] Technically, Grömping [248] is an updated and corrected version of the dataset from 2019.

[10] `https://archive.ics.uci.edu/ml/datasets/South+German+Credit+%28UPDATE%29`

and process the data ourselves. We use the provided `codetable.txt` to "translate" the features from German to English. We say "translate" because the authors took some liberties, e.g., the column converted to "credit_history" is labeled "moral" in the German, which is not a translation. There are four categories in the protected attribute "personal_status_sex" column, one of which (2) is used for both "Male (single)" and "Female (non-single)." We therefore remove rows with this value, and binarize the remaining three categories into "Female" (0) and "Male" (1). What results is a dataset with 690 example instances (of the original 1000) with 19 features. The target is "good" credit (1) and "bad" credit (0).

- `Taiwan Credit` [630]. This task is to predict default on credit card payments (1) or not (0). There are 30,000 example instances and 24 features. The protected attribute is binary `sex`. We download this dataset from UCI.[11].

- `New Adult` [174]. This dataset contains millions of example instances from US Census data, which can be used for several different targets/-tasks. We select three of them (listed below). These tasks share some features, and therefore are not completely independent. Further, given the size of the whole dataset, we subset to `CA` (California), the most populous state in the US. There are two protected attribute columns that we use: `sex`, which is binarized "Female" (0) and "Male" (1) subgroups, and `race`, which we binarize into "Non-white" (0) and "White" (1). In future work, we would like to explore extending our results beyond binary subgroups.

  - `Income`. This task is designed to be analogous to `Old Adult` [322].

[11] See `https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients`

As a result, the target is to predict < $50,000 income (0) >= $50,000 income (1). In the CA subset, there are 195,665 example instances with 8 features.

- Employment. This task is to predict whether an individual is employed (1) or not (0). In the CA subset, there are 378,817 example instances with 14 features.
- Public Coverage. This task is to predict whether an individual is on public health insurance (1) or not (0). In the CA subset, there are 138,554 example instances with 17 features.

The standalone HMDA tookit

In addition to the above standard tasks, we include experiments that use the NY and TX 2017 subsets of the the Home Mortgage Data Disclosure Act (HMDA) 2007-2017 dataset [206]. These two datasets have 244,107 and 576,978 examples, respectively, with 18 features. The HMDA datasets together contain over 140 million examples of US home mortgage loans from 2007-2017 (newer data exists, but in a different format). We developed a toolkit, described below, to make this dataset easy to use for classification experiments. Similar to New Adult, we enable subsetting by US state. For the experiments in this paper, we run on the NY (New York) and TX (Texas) 2017 subset, in order to add some geographic diversity to complement our New Adult experiments. We additionally chose NY and TX because they are two of the most populous states in the US, alongside CA.[12]

The target variable, action_taken, concerning loan origination has 8 val-

[12]Per the 2020 Census, the top-4-most-populous states are CA, TX, FL, and NY [389].

ues, 2 of which we cannot meaningful conclude approval or denial decisions. They are: Action Taken: 1 – Loan originated, 2 – Application approved but not accepted, 3 – Application denied by financial institution, 4 – Application withdrawn by applicant, 5 – File closed for incompleteness, 6 – Loan purchased by the institution, 7 – Preapproval request denied by financial institution, and 8 – Preapproval request approved but not accepted (optional reporting). We filter out 4 and 6, and binarize into `grant`$=\{1,2,8\} = 1$ and `reject`$=\{3,5,7\} = 0$. There are three protected attributes that we consider: `sex`, `race`, and `ethnicity`:

- `sex` has 5 possible values, 2 of which correspond to categories/non-missing values: Male – 1 and Female – 2. We binarize `sex` into F = 0 and M = 1.
- `race` has 8 possible values, 5 of which correspond to categories/ non-missing information: 1 – American Indian or Alaska Native, 2 – Asian, 3 – Black or African American, 4 – Native Hawaiian or Other Pacific Islander, and 5 – White. There are 5 fields for applicant race, which model an applicant belonging to more than one racial group. For our experiments, we only look at the first field. When we binarize `race`, NW = 0 and W = 1.
- `ethnicity` has 5 possible values, 2 of which correspond to categories/ non-missing information: 1 – Hispanic or Latino and 2 – Not Hispanic or Latino. We binarize ethnicity to be HL = 0 and NHL = 1.

After subsetting to only include examples that have values that do not correspond to missing information, `HMDA` has 18 features. The `NY` dataset has 244,107 examples; the `TX` dataset has 576,978 examples, making it the largest dataset in our experiments. As with our experiments using `New Adult`, we would like to

extend our results beyond binary subgroups and binary classification in future work.

Releasing a standalone toolkit. These datasets are less-commonly used in current algorithmic fairness literature [194]. We believe this is likely due to the fact that the over-100-million data examples are only available in bulk files, which are on the order of 10s of gigabytes and therefore not easily downloadable or explorable on most personal computers. Following the example of Ding et al. [174], one of our contributions is to pre-process all of these datasets — all locations and years — and release them with a software toolkit. The software engineering effort to produce this toolkit was substantial. Our hope is that wider access to this dataset will further reduce the community's dependency on small (and dated) datasets. Please refer to the arXiv paper for the latest information on this standalone software package. Our release aligns with the terms of service for this dataset.

C.5.2 Cluster environment details

While most of the experiments run in this paper can be easily reproduced on a modern laptop, for efficiency, we ran all of our experiments (except the one to produce Figure 3.1) in a cluster environment. This enabled us to easily execute train/test splits n in parallel on different CPUs, serialize our results, and then reconstitute and combine them to produce plots locally. Our cluster environment runs Ubuntu 20.04 and uses Slurm v20.11.8 to manage jobs. We ran all experiments using `Anaconda3`, which is why we used `Conda` to reproduce environments for easy replicability.

The experiments using `New Adult` and `HMDA` rely on datasets that are (in some cases) orders of magnitude larger than the traditional algorithmic fairness tasks. This is one of the reasons why we recommend running on a cluster, and therefore do not include Jupyter notebooks in our repository for these tasks. We also limit our modeling choices to logistic regression, decision tree classifiers, and random forest classifiers for these results due to the expense of training on the order of thousands of models for each experiment.

C.5.3 Details on motivating examples in the main paper

This appendix provides extended results for the experiments associated in Sections 3.1 and 3.3, which give an intuition for individual- and subgroup-level consistency. The experimental results in the main paper are for logistic regression. We expand the set of models we examine, and associated discussion of how to interpret comparisons between these results.

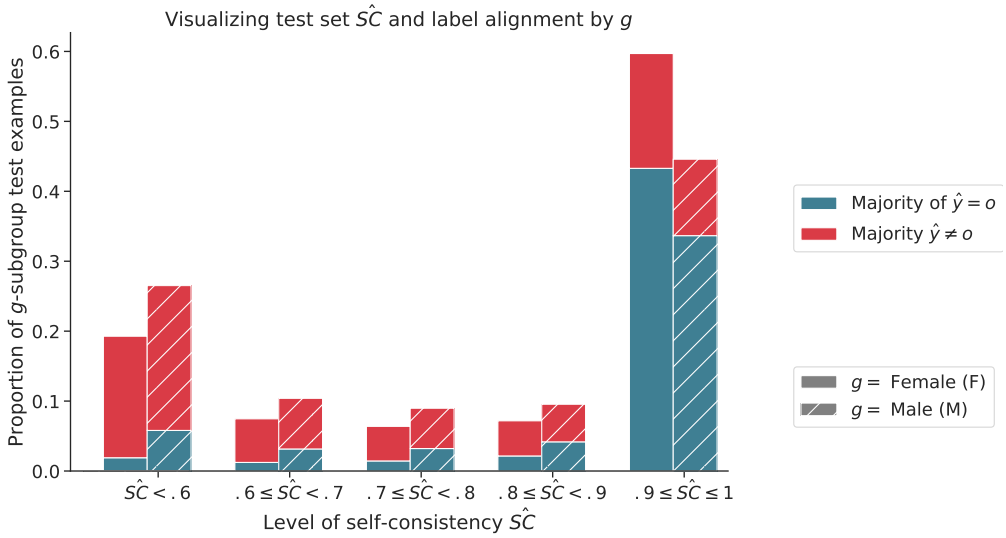
Reproducing Figure 3.1. The experiment to produce this figure in Section 3.1 (also shown in Appendix C.2.3) trains $B = 10$ logistic regression models on the `COMPAS` dataset (Appendix C.5.1) using 0-1 loss. We use the bootstrap method to produce each model, which we evaluate on the same test set. We then search for a maximally consistent and minimally consistent individual in the test set, i.e., an individual with 10 predictions that agree and an individual with 5 predictions in each class, which we plot in the bar graph. Please refer to the README in the code repository (see arXiv paper) regarding which `Jupyter` notebook to run to produce the underlying results and figure. The experiments to reproduce this figure can be easily replicated on a laptop.

Reproducing Figure 3.2. These figures were produced by executing $S = 10$ runs of $B = 101$ bootstrap training replicates to train random forest classifiers for `Old Adult` and `COMPAS`. We reproduce these figures below, so that they can be examined and treated in relation to our additional results for decision tree classifiers and logistic regression. For each s run, we take train/test split, bootstrap the train split $B = 101$ times, and evaluate the resulting model classification decisions on the test set. $\widehat{\texttt{SC}}$ can be estimated from the results across those 101 models. We Run this process $S = 10$ times to produce confidence intervals, shown in the figures below. The intervals are not always clearly visible; there is not a lot of variance at the level of comparing whole runs to each other. Please refer to the README in the code repository (see arXiv paper) regarding which `Jupyter` notebook to run to produce the underlying results and figure. There are also scripted version of these experiments, which enable them to be run in parallel in a cluster environment.

Self-consistency of incorrectly-classified instances. Last, we include figures that underscore how self-consistency is independent from correctness that is measured in terms of observed label alignment. That is, it is possible for an instance $(\boldsymbol{x}, \boldsymbol{g})$ to be self-consistent and classified incorrectly, with respect to its observed label o. We show this using stacked bar plots. For the above experiments, we find the test examples that have the majority of their classifications incorrect ($\hat{y} \neq o$, for $B = 101$, we find the instances with ≥ 51 incorrect classifications) and the majority of their classification correct (similarly), and we examine how self-consistent they are. We bucket self-consistency into different levels, and then plot the relative proportion of majority-incorrectly and majority-correctly classified examples according to subgroup. Subgroups in `COMPAS` exhibit a similar

trend, while subgroups in `Adult Old` exhibit differences, with the heights of the bars corresponding to the trends we plot in our CDF plots. As we note briefly in Section 3.3, it may be interesting to examine patterns in examples about which learning processes are confident (i.e., highly self-consistent) but wrong in terms of label alignment. If such issues correlate with subgroup, it may be worth testing the counterfactual that such labels are indicative of label bias. We leave such thoughts to future work.

(a) COMPAS



(b) Adult Old

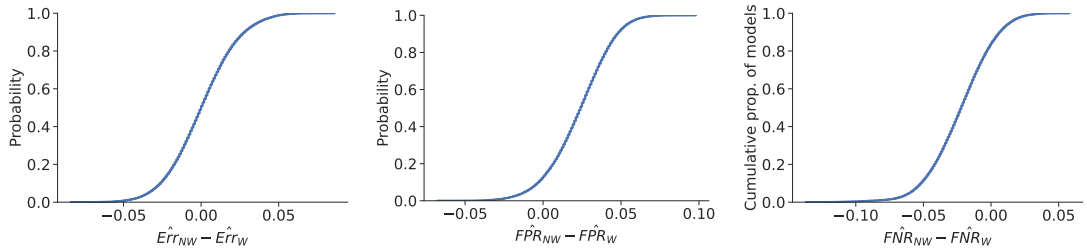
Figure C.1: $\hat{\text{SC}}$ broken down by $\boldsymbol{g}$ and label alignment with the observed label o. For each train/test split, and for each $\hat{\text{SC}}$ range (x-axis), we find the examples that are incorrectly classified the majority of time (≥ 5 splits, we find that $\hat{y} \neq o$), and the examples that are correctly classified the majority of the time (> 5, we find that $\hat{y} = o$). We compute the average the proportion over (over splits) in each $\hat{\text{SC}}$ range (y-axis). We plot these proportions with respect to subgroup $\boldsymbol{g}$ (where the sums of the heights of bars for by each $\boldsymbol{g}$ is equal to 1).

C.5.4 Reliability and fairness metrics in `COMPAS` and `South German Credit`

Even before we apply our intervention to improve self-consistency, our results in Section 3.3 show close-to-parity $\hat{\texttt{Err}}$, $\hat{\texttt{FPR}}$, and $\hat{\texttt{FNR}}$ across subgroups in `COMPAS` (and similarly for `South German Credit`, below). These results are surprising. We run $B = 101$ models to produce estimates of variance and self-consistency, but of course doing this also has the effect of estimating the expected error more generally (with variance representing a portion of that error).

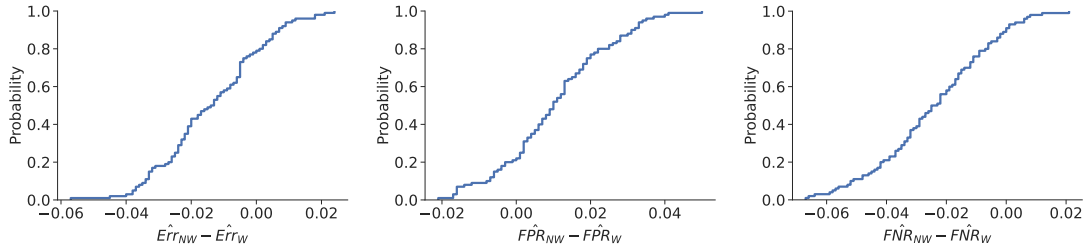
Our estimates of expected error for these tasks indicate that the average model produced training on `COMPAS` and `South German Credit`, with respect to popular fairness definitions like Equality of Opportunity and Equalized Odds [45, 260] are in fact baseline close to parity, with no fairness intervention applied. We found this across model types for both datasets, though the story becomes more complicated when we apply techniques to improve self-consistency (see arXiv appendix).

We did not expect this result, as these are two of the *de facto* standard benchmark datasets in algorithmic fairness. They are used in countless other studies to probe and verify algorithmic fairness interventions [194]. As a result, we initially thought that our results must be incorrect. We therefore looked at the underlying models in our bootstrap runs to see the error of the underlying models. We re-ran our baseline experiments with $B = 1001$ and for 100 test/-train splits for logistic regression. In Figure C.2, we plot the $(100, 100)$ bootstrap models that went into these results. For another view on analogous information, in Table C.2, we provide an excerpt of the results for `COMPAS` regarding the



(a) $\hat{\texttt{Err}}$ disparity (b) $\hat{\texttt{FPR}}$ disparity (c) $\hat{\texttt{FNR}}$ disparity

Figure C.2: Cumulative distribution of error disparity across 100, 100 logistic regression models trained on COMPAS.



(a) $\hat{\texttt{Err}}$ disparity (b) $\hat{\texttt{FPR}}$ disparity (c) $\hat{\texttt{FNR}}$ disparity

Figure C.3: CDF of error disparity across the top 100 logistic regression models (of the 100, 100 models) trained on COMPAS.

underlying 1010 random forest classifiers used to produce Figure 3.2a.

Overall, we can see that there is a wide range of error disparities that trend in both directions, with a skew toward higher $\hat{\texttt{FPR}}$ for $\boldsymbol{g}$ = NW. These results support our claim that training many models is necessary to get an accurate picture of expected error, with implications both for reproducibility of experiments that just train and analyze a small handful of models and for generalizability. There are models that exhibit worse degrees of unfairness in both directions, but they are more unlikely than models that exhibit smaller disparities.

We subset the above results to the 100 models that produce the lowest $\hat{\texttt{Err}}$, as this is often the selection criteria for picking models to post-process. We plot these results below. These top-performing models in fact exhibit (on average) closer-to-parity for $\hat{\texttt{FPR}}$ and $\hat{\texttt{FNR}}$.

Table C.2: Comparing subgroup error rates in COMPAS for different random forest classifiers trained to produce Figure 3.2a. Each table looks at the top-3 highest differences between subgroups for the specified metric: (a) $\widehat{\text{Err}}_{\text{NW}} - \widehat{\text{Err}}_{\text{W}}$, when $\widehat{\text{Err}}_{\text{NW}} > \widehat{\text{Err}}_{\text{W}}$; (b) $\widehat{\text{Err}}_{\text{W}} - \widehat{\text{Err}}_{\text{NW}}$, when $\widehat{\text{Err}}_{\text{W}} > \widehat{\text{Err}}_{\text{NW}}$; (c) $\widehat{\text{FPR}}_{\text{NW}} - \widehat{\text{FPR}}_{\text{W}}$, when $\widehat{\text{FPR}}_{\text{NW}} > \widehat{\text{FPR}}_{\text{W}}$; (d) $\widehat{\text{FPR}}_{\text{W}} - \widehat{\text{FPR}}_{\text{NW}}$, when $\widehat{\text{FPR}}_{\text{W}} > \widehat{\text{FPR}}_{\text{NW}}$; (e) $\widehat{\text{FNR}}_{\text{NW}} - \widehat{\text{FNR}}_{\text{W}}$, when $\widehat{\text{FNR}}_{\text{NW}} > \widehat{\text{FNR}}_{\text{W}}$; and, (f) (e) $\widehat{\text{FNR}}_{\text{W}} - \widehat{\text{FNR}}_{\text{NW}}$, when $\widehat{\text{FNR}}_{\text{W}} > \widehat{\text{FNR}}_{\text{NW}}$. We highlight the overall error metric in gray, the larger metric (being subtracted from) in blue, the smaller metric (being subtracted) in red, and the difference in the metric between subgroups in purple. Note that run 757 appears twice, which we mark in orange.

(a) Top-3 most unfair models by $\widehat{\text{Err}}$, when $\widehat{\text{Err}}_{\text{NW}} > \widehat{\text{Err}}_{\text{W}}$ (i.e., unfair toward NW).

Run #	s	b	$\widehat{\text{Err}}$	$\widehat{\text{FPR}}$	$\widehat{\text{FNR}}$	$\widehat{\text{Err}}_{\text{NW}}$	$\widehat{\text{FPR}}_{\text{NW}}$	$\widehat{\text{FNR}}_{\text{NW}}$	$\widehat{\text{Err}}_{\text{W}}$	$\widehat{\text{FPR}}_{\text{W}}$	$\widehat{\text{FNR}}_{\text{W}}$	$\widehat{\text{Err}}_{\text{NW}} - \widehat{\text{Err}}_{\text{W}}$
762	8	504	0.374	0.179	0.196	0.405	0.204	0.201	0.315	0.13	0.186	0.09
757	8	464	0.369	0.167	0.202	0.395	0.201	0.193	0.318	0.101	0.218	0.077
328	4	116	0.371	0.165	0.206	0.395	0.181	0.214	0.323	0.134	0.189	0.072

(b) Top-3 most unfair models by $\widehat{\text{Err}}$, when $\widehat{\text{Err}}_{\text{W}} > \widehat{\text{Err}}_{\text{NW}}$ (i.e., unfair toward W).

Run #	s	b	$\widehat{\text{Err}}$	$\widehat{\text{FPR}}$	$\widehat{\text{FNR}}$	$\widehat{\text{Err}}_{\text{NW}}$	$\widehat{\text{FPR}}_{\text{NW}}$	$\widehat{\text{FNR}}_{\text{NW}}$	$\widehat{\text{Err}}_{\text{W}}$	$\widehat{\text{FPR}}_{\text{W}}$	$\widehat{\text{FNR}}_{\text{W}}$	$\widehat{\text{Err}}_{\text{W}} - \widehat{\text{Err}}_{\text{NW}}$
414	5	75	0.376	0.167	0.209	0.352	0.158	0.194	0.422	0.186	0.236	0.07
435	5	180	0.376	0.199	0.177	0.355	0.189	0.166	0.416	0.217	0.198	0.061
413	5	70	0.378	0.189	0.189	0.359	0.188	0.171	0.413	0.191	0.222	0.054

(c) Top-3 most unfair models by $\widehat{\text{FPR}}$, when $\widehat{\text{FPR}}_{\text{NW}} > \widehat{\text{FPR}}_{\text{W}}$ (i.e., unfair toward NW).

Run #	s	b	$\widehat{\text{Err}}$	$\widehat{\text{FPR}}$	$\widehat{\text{FNR}}$	$\widehat{\text{Err}}_{\text{NW}}$	$\widehat{\text{FPR}}_{\text{NW}}$	$\widehat{\text{FNR}}_{\text{NW}}$	$\widehat{\text{Err}}_{\text{W}}$	$\widehat{\text{FPR}}_{\text{W}}$	$\widehat{\text{FNR}}_{\text{W}}$	$\widehat{\text{FPR}}_{\text{NW}} - \widehat{\text{FPR}}_{\text{W}}$
757	8	464	0.369	0.167	0.202	0.395	0.201	0.193	0.318	0.101	0.218	0.1
729	8	240	0.358	0.162	0.197	0.376	0.189	0.187	0.323	0.107	0.216	0.082
791	8	736	0.377	0.171	0.205	0.395	0.198	0.197	0.341	0.118	0.222	0.08

(d) Top-3 most unfair models by $\widehat{\text{FPR}}$, when $\widehat{\text{FPR}}_{\text{W}} > \widehat{\text{FPR}}_{\text{NW}}$ (i.e., unfair toward W).

Run #	s	b	$\widehat{\text{Err}}$	$\widehat{\text{FPR}}$	$\widehat{\text{FNR}}$	$\widehat{\text{Err}}_{\text{NW}}$	$\widehat{\text{FPR}}_{\text{NW}}$	$\widehat{\text{FNR}}_{\text{NW}}$	$\widehat{\text{Err}}_{\text{W}}$	$\widehat{\text{FPR}}_{\text{W}}$	$\widehat{\text{FNR}}_{\text{W}}$	$\widehat{\text{FPR}}_{\text{W}} - \widehat{\text{FPR}}_{\text{NW}}$
639	7	280	0.36	0.187	0.173	0.352	0.174	0.178	0.376	0.212	0.164	0.038
807	9	72	0.381	0.191	0.19	0.372	0.179	0.192	0.398	0.214	0.184	0.035
543	6	264	0.358	0.155	0.203	0.351	0.144	0.206	0.37	0.175	0.196	0.031

(e) Top-3 most unfair models by $\widehat{\text{FNR}}$, when $\widehat{\text{FNR}}_{\text{NW}} > \widehat{\text{FNR}}_{\text{W}}$ (i.e., unfair toward NW).

Run #	s	b	$\widehat{\text{Err}}$	$\widehat{\text{FPR}}$	$\widehat{\text{FNR}}$	$\widehat{\text{Err}}_{\text{NW}}$	$\widehat{\text{FPR}}_{\text{NW}}$	$\widehat{\text{FNR}}_{\text{NW}}$	$\widehat{\text{Err}}_{\text{W}}$	$\widehat{\text{FPR}}_{\text{W}}$	$\widehat{\text{FNR}}_{\text{W}}$	$\widehat{\text{FNR}}_{\text{NW}} - \widehat{\text{FNR}}_{\text{W}}$
246	3	141	0.379	0.166	0.213	0.398	0.169	0.229	0.345	0.161	0.184	0.045
506	6	42	0.367	0.17	0.197	0.386	0.175	0.211	0.332	0.161	0.171	0.04
204	3	15	0.384	0.185	0.199	0.394	0.181	0.213	0.365	0.192	0.173	0.04

(f) Top-3 most unfair models by $\widehat{\text{FNR}}$, when $\widehat{\text{FNR}}_{\text{W}} > \widehat{\text{FNR}}_{\text{NW}}$ (i.e., unfair toward W).

Run #	s	b	$\widehat{\text{Err}}$	$\widehat{\text{FPR}}$	$\widehat{\text{FNR}}$	$\widehat{\text{Err}}_{\text{NW}}$	$\widehat{\text{FPR}}_{\text{NW}}$	$\widehat{\text{FNR}}_{\text{NW}}$	$\widehat{\text{Err}}_{\text{W}}$	$\widehat{\text{FPR}}_{\text{W}}$	$\widehat{\text{FNR}}_{\text{W}}$	$\widehat{\text{FNR}}_{\text{W}} - \widehat{\text{FNR}}_{\text{NW}}$
474	5	375	0.373	0.175	0.199	0.356	0.183	0.174	0.406	0.159	0.247	0.073
401	5	10	0.378	0.189	0.19	0.363	0.197	0.167	0.406	0.173	0.233	0.066
52	1	53	0.367	0.172	0.196	0.351	0.178	0.173	0.397	0.16	0.238	0.065

This detailed view provides insight into how such a result is possible. Broadly speaking, individual runs have roughly similar error;[13] yet, the subgroup-specific error rates that compose the overall error can nevertheless vary widely depending on the underlying training data. This observation aligns with current interest in *model multiplicity* in the algorithmic fairness community [68, 616], which imports the idea from Breiman [86]. In this case, as suggested by Table C.2, there are models that demonstrate unfairness toward both subgroups with respect to each error rate metric $\texttt{E}\hat{\texttt{r}}\texttt{r}$, $\texttt{F}\hat{\texttt{P}}\texttt{R}$, and $\texttt{F}\hat{\texttt{N}}\texttt{R}$. When we move away from attempting to find a *single* model that performs well (accurately or fairly) on COMPAS, and instead consider the information contained across different possible models, we yield the result that the average, expected behavior smooths over the variance in underlying models such that the result is close to fair.

Stability analysis. To verify the stability of this result, we re-execute our experiments for increasing numbers of train/test splits S and replicates B. While our results for COMPAS are generally tight for small S (e.g., Figures 3.2a and 3.5), this was not the case for German Credit, for which it was difficult to estimate self-consistency consistently. As a result, for COMPAS, we did not expect markedly different results for increased S. Our results for $S = 100, B = 1001$ using logistic regression (Figure C.4, Table C.3) confirm this intuition.

[13] This should be taken relatively. In general, COMPAS demonstrates high error; the error is relatively tight given just how much error there is. The error fluctuates depending on the training data, but the average error rate across train/test splits is rather tight, despite the fluctuations in error within the B runs of each split.

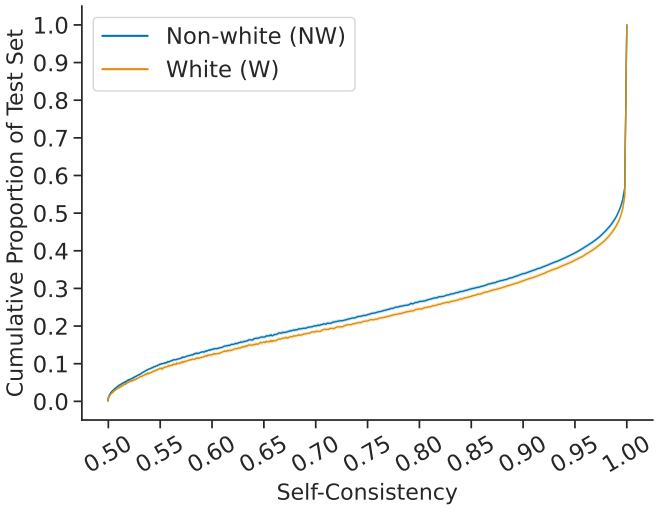


Figure C.4: `COMPAS` split by $\boldsymbol{g}$ = `race`, $B = 1001, S = 100$

Table C.3: Mean ± STD across $S = 100$ train/test splits $\times$ $B = 1001$ runs.

	COMPAS			
	$\widehat{\textbf{Err}}$	$\widehat{\textbf{FPR}}$	$\widehat{\textbf{FNR}}$	$\widehat{\textbf{SC}}$
Total	.333 ± .008	.14 ± .009	.192 ± .01	.883 ± .004
g = NW	.333 ± .01	.148 ± .011	.185 ± .012	.88 ± .005
g = W	.332 ± .014	.125 ± .013	.207 ± .016	.888 ± .006

We provide analogous results for `German Credit`, with $S = 1000, B = 1001$ using random forests (Figure C.5, Table C.4). It takes an enormous number of runs to produce stable estimates of error and $\widehat{\text{SC}}$ for `German Credit`, which indicate statistical equality across groups. Arguably, our results below for $1,001,000$ models still are very high variance (certainly with respect to error metrics). This task really has too few data points (≈ 600) to generalize reliably.

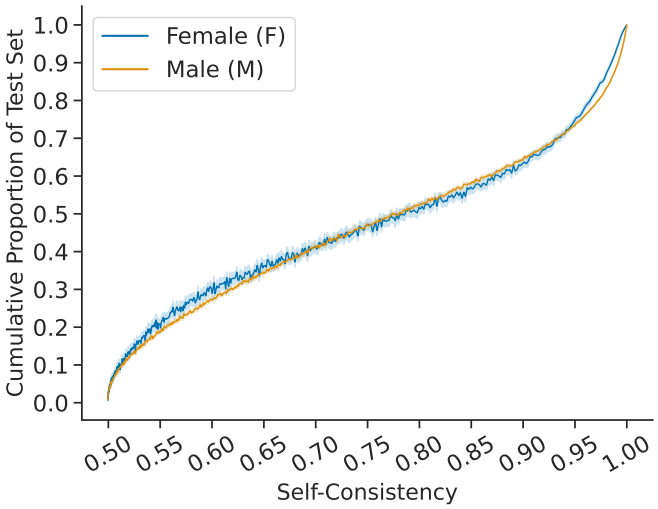


Figure C.5: `German Credit` split by $\boldsymbol{g}$ = `sex`, $S = 1000$, $B = 100$

Table C.4: Mean ± STD across $S = 1000$ train/test splits × $B = 1001$ runs.

South German Credit				
	$\widehat{\textbf{Err}}$	$\widehat{\textbf{FPR}}$	$\widehat{\textbf{FNR}}$	$\widehat{\textbf{SC}}$
Total	.28 ± .021	.173 ± .028	.107 ± .017	.769 ± .015
g = F	.288 ± .064	.183 ± .072	.105 ± .037	.766 ± .04
g = M	.279 ± .023	.171 ± .029	.108 ± .018	.769 ± .016

C.6 Brief notes on future research

There are many interesting directions for future work.

Novel theory. We do not include extensive novel theory in this project. Nevertheless, our project raises interesting questions for theory in future work. Notably, we could compose our methodology with post-processing [260] for cases in which there is observed empirical unfairness. We could then investigate picking group-specific thresholds that take variance into account. We could recon-

figure the formulations in [260] and related work, with respect to the fairness-accuracy trade-off, as actually representing multiple such trade-off curves (that are a function of different models under consideration). There may be interesting directions for mathematical analysis in this direction.

We could also extend traditional results on bagging and variance reduction for classifiers. While bagging has guarantees for variance reduction for regression, it does not have the same guarantees for classification [84, 85]. It generally is observed to work well in practice for variance reduction if the underlying classifiers are high variance — which is indeed the regime we are in for this paper. However, there are interesting theory questions regarding abstention that we could investigate with theoretical tools, which could let us come up with other ways of reasoning about bagging and variance reduction.

Both of these directions are out of scope for the present paper. They are interesting, but do not have to do with our main experimental aims and contributions, and thus do not make it into a conference-length submission. We are not interested in novel theory in the present study. If anything, our work highlights how over-attention to theory can (directly or indirectly) bring about serious problems of mismeasurement in practice. That is a main takeaway for our work, which by nature does not involve novel theory.

Arbitrariness beyond algorithmic fairness. Our framework for reasoning about self-consistency and arbitrariness does not inherently have to do with algorithmic fairness. We could apply it to other domains. For example, it would be interesting to ask similar questions in deep learning and generative AI. We think that such work would be interesting, but is again out of scope for the

present study. The first author of this project is in fact working on such questions as separate work. However, this project's research aims are inherently focused on fairness; the project was designed in response to observations in experimental practices in the fairness community, fairness definitions, and fairness theory.

Experiments on synthetic data. Our results indicate that unfairness (as defined with respect to model error rates) is not frequently observed on common benchmark tasks in fair classification. Of course, there could be other datasets in fairness domains that are not currently used as benchmarks that more clearly demonstrate unfairness in practice. Hypothetically, there could be datasets for which we use Algorithm 2 to reduce arbitrariness, and yet we still see significant systematic arbitrariness or differences in error rates (and thus unfairness) due to noise or bias. We just did not really see this for almost all of the tasks we investigate in this paper, which happen to be the ones that the fairness community uses for experiments.

To study Algorithm 2 in light of these other possibilities, we could develop synthetic datasets that retain unfairness after dealing with arbitrariness. We did not do this in the present study for two reasons. First, our focus was the practice of fairness research, as it currently stands, with a data-centric approach on the datasets people actually use for their research. We are not interested in synthetic data for this project. However, future theory results that extend our work could be vetted experimentally with synthetic data. The work we mention above regarding composition with post-processing, as well as revisting impossibility results from a distributional approach over possible models, may be very interesting to examine under data settings that we can control.

How to deal with abstention. Future work could also perform a deeper exploration of the trade-off between abstention rate and error. We could characterize a Pareto-optimal trade-off that is a function of the choice of self-consistency level κ, and also examine in experiments and analytically how abstention leads to improvements in accuracy. Future work could also identify patterns in abstention sets beyond low self-consistency. In this, looking to metrics from model multiplicity may be helpful. Further, future work could combine human decision-making or other automated elements to see how we can root out arbitrariness.

Reproducibility. As mentioned in our Ethics Statement, we made attempts to reproduce prior work in fair classification, and often could not. We ultimately made reproducibility of specific papers out of scope for the present project, as we could make our contributions about arbitrariness and variance without such work. It would nevertheless be useful to focus future work on reproducing prior algorithmic fairness studies, and seeing if conclusions change in those works as a function of using Algorithm 2 prior to introducing the proposed fairness intervention.

Law and policy. Our work regarding arbitrariness raises concrete questions for the law around due process and automated decision-making. Preliminary exploration of these ideas can be found in Cooper et al. [138] (Chapter 4). Developing further contributions in this line of work is also out of the scope of the present work. We are currently pursuing this work for a future submission to a law review journal.

APPENDIX D

APPENDIX FOR TUNAMH

D.1 Proof of Theorem 2

In this section, we prove Theorem 2, which asserts that any inexact stateless MH algorithm can produce arbitrarily large bias between its target distribution (the distribution we are trying to sample from) and its stationary distribution (the distribution that the chain actually produces samples from asymptotically).

Proof. Let $\mathcal{A}$ denote the `SubsMH` in Algorithm 3 of the minibatch MH method in question. Since $\mathcal{A}$ is inexact, there must exist a state space Θ, proposal distribution q, and target distribution μ, satisfying Assumption 1 with parameters $c_1, \ldots, c_N, C, M$, where

$$\mu(\theta) \propto \exp\left(-\sum_{i=1}^{N} V_i(\theta)\right)$$

for some N and energy functions $V_1, \ldots, V_N$, such that $\mathcal{A}$ run on μ with proposal distribution q does not have stationary distribution μ.

Next, let $a_\mu(\theta, \theta')$ denote the acceptance probability of algorithm $\mathcal{A}$ on the above task for a proposed transition from θ to θ'. Assume by way of contradiction that on this problem, it is always true that

$$\frac{a_\mu(\theta, \theta')}{a_\mu(\theta', \theta)} = \frac{\mu(\theta')q(\theta|\theta')}{\mu(\theta)q(\theta'|\theta)}.$$

If this were true, then the overall transition probability of this chain, for $\theta \neq \theta'$, would be

$$T_\mu(\theta, \theta') = q(\theta'|\theta) \cdot a_\mu(\theta, \theta')$$

and it would hold that

$$\mu(\theta)T_\mu(\theta,\theta') = \mu(\theta')T_\mu(\theta',\theta).$$

That is, the chain would be reversible, also known as satisfying detailed balance. But it is a standard result that for any reversible chain, μ must be a stationary distribution of that chain. We have now derived a contradiction, which establishes that our assumption is false. That is, there exists a $\theta, \theta' \in \Theta$ such that

$$\frac{a_\mu(\theta,\theta')}{a_\mu(\theta',\theta)} \neq \frac{\mu(\theta')\cdot q(\theta|\theta')}{\mu(\theta)\cdot q(\theta'|\theta)}.$$

Explicitly, this means that if we define the function ΔV such that

$$\Delta V(i) = V_i(\theta) - V_i(\theta'),$$

then for this subsampling problem,

$$\frac{\mathbf{E}\left[\mathcal{A}(\Delta V, N, q(\theta|\theta')/q(\theta'|\theta), c_1, \ldots, c_N, C, M(\theta,\theta'))\right]}{\mathbf{E}\left[\mathcal{A}(-\Delta V, N, q(\theta'|\theta)/q(\theta|\theta'), c_1, \ldots, c_N, C, M(\theta,\theta'))\right]} \neq \frac{\mu(\theta')\cdot q(\theta|\theta')}{\mu(\theta)\cdot q(\theta'|\theta)}. \tag{D.1}$$

Without loss of generality, assume that

$$q(\theta|\theta')/q(\theta'|\theta) \leq 1.$$

(This is without loss of generality since we can ensure it is the case by swapping θ and θ'.) We fixed θ and θ' to be the pair satisfying Equation D.1 throughout this section.

Constructing an example. We use this to prove the theorem by a constructive example. Let $x_1, \ldots, x_N$ be defined by

$$x_i = \Delta V(i) = V_i(\theta) - V_i(\theta').$$

Define X as the sum

$$X = \sum_{i=1}^{N} x_i.$$

For some parameter $K \in \mathbb{N}$ (to be defined later), consider the state space Ω defined as

$$\Omega = \{(k, z) \mid k \in \{0, \ldots, K-1\},\ 0 \leq z \leq \exp(kX)\},$$

using the natural measure for a finite disjoint union of measure spaces. Define a target distribution over Ω given by the density

$$\pi(k, z) \propto \exp\left(-\sum_{i=1}^{N} k \cdot x_i\right),$$

or equivalently

$$\pi(k, z) \propto \exp\left(-\sum_{i=1}^{N} U_i(k, z)\right) \text{ where } U_i(k, z) = kx_i.$$

Define a proposal distribution $\hat{q}$, such that, starting from (k, z):

- With probability $1/4$, we sample z' uniformly from $[0, \exp(kX)]$ and propose a transition to (k, z').
- With probability $1/4$, we propose a transition to $(k-1, z)$, if it is in Ω.
- With probability $\frac{1}{4} \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}$, we propose a transition to $(k+1, z)$, if it is in Ω.
- With the remaining probability, we just propose to stay at (k, z).

This is effectively acting as a random walk over k, and our goal will be to show that while the true target distribution π has a marginal in k that is the uniform distribution, the minibatch MH method causes the chain's transition to be biased to step more in one direction than another, resulting in a highly biased stationary distribution (where we can make the bias arbitrarily large by setting K).

We use the same c_i and C as before, and define a new function $\hat{M}$ such that

$$\hat{M}((k, z), (k+1, z)) = \hat{M}((k, z), (k-1, z)) = M(\theta, \theta')$$

and $\hat{M}(\cdots) = 0$ for other proposed transitions (we can set $\hat{M}$ however we want for pairs of states that are never proposed in a transition, since this will not affect the algorithm). Clearly, this setup satisfies Assumption 1, since the original distribution did.

Now, consider what our minibatch MH method will do when run on this task. There are three cases to consider.

Proposed changes in z. When a proposed change in z is made, the resulting ΔU will be uniformly 0, and the probability of the reverse transition will be equal (1/4 in both directions), so the algorithm will be passed the arguments

$$\mathcal{A}(0, N, 1, c_1, \ldots, c_N, C, 0).$$

Since this does not depend at all on z or k, this means that the acceptance probability of these transitions will be the same regardless of the state. Call this probability α_0.

A proposal to decrease k. When a proposal is made to decrease k, the probability of the forward and reverse transitions will be

$$\hat{q}((k-1,z)|(k,z)) = \frac{1}{4} \text{ and } \hat{q}((k,z)|(k-1,z)) = \frac{1}{4} \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}.$$

It follows that

$$\frac{\hat{q}((k,z)|(k-1,z))}{\hat{q}((k-1,z)|(k,z))} = \frac{q(\theta|\theta')}{q(\theta'|\theta)}.$$

The energy function difference for this proposal will be

$$\Delta U(i) = U_i((k,z)) - U_i((k-1,z)) = kx_i - (k-1)x_i = x_i,$$

so in particular $\Delta U = \Delta V$. And, of course for this transition $\hat{M}$ will take on the value $M(\theta, \theta')$. So, the minibatch MH algorithm will be passed the arguments

$$\mathcal{A}(\Delta V, N, q(\theta|\theta')/q(\theta'|\theta), c_1, \ldots, c_N, C, M(\theta, \theta')),$$

and so it will accept with probability

$$\mathbf{E}\left[\mathcal{A}(\Delta V, N, q(\theta|\theta')/q(\theta'|\theta), c_1, \ldots, c_N, C, M(\theta, \theta'))\right].$$

Call this probability α_-.

A proposal to increase k. When a proposal is made to increase k, the probability of the forward and reverse transitions will be

$$\hat{q}((k+1, z)|(k, z)) = \frac{1}{4} \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}. \text{ and } \hat{q}((k, z)|(k+1, z)) = \frac{1}{4}.$$

It follows that

$$\frac{\hat{q}((k, z)|(k+1, z))}{\hat{q}((k+1, z)|(k, z))} = \frac{q(\theta'|\theta)}{q(\theta|\theta')}.$$

The energy function difference for this proposal will be

$$\Delta U(i) = U_i((k, z)) - U_i((k+1, z)) = kx_i - (k+1)x_i = -x_i,$$

so in particular $\Delta U = -\Delta V$. And, as before for this transition $\hat{M}$ will take on the value $M(\theta, \theta')$. So, the minibatch MH algorithm will be passed the arguments

$$\mathcal{A}(-\Delta V, N, q(\theta'|\theta)/q(\theta|\theta'), c_1, \ldots, c_N, C, M(\theta, \theta')),$$

and so it will accept with probability

$$\mathbf{E}\left[\mathcal{A}(-\Delta V, N, q(\theta'|\theta)/q(\theta|\theta'), c_1, \ldots, c_N, C, M(\theta, \theta'))\right].$$

Define the probability α_+ as

$$\alpha_+ = \mathbf{E}\left[\mathcal{A}(-\Delta V, N, q(\theta'|\theta)/q(\theta|\theta'), c_1, \ldots, c_N, C, M(\theta, \theta'))\right] \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}.$$

The resulting Markov chain. From the above analysis, we can conclude that the Markov chain that results from subsampling algorithm $\mathcal{A}$ applied to this method is as follows. Starting from (k, z), if we let $\hat{T}$ denote the transition operator of this Markov chain,

- With probability $\frac{1}{4} \cdot \alpha_0$, we sample z' uniformly from $[0, \exp(kX)]$ and transition to (k, z').
- With probability $\frac{1}{4} \cdot \alpha_-$, we transition to $(k-1, z)$, if it is in Ω.
- With probability $\frac{1}{4} \cdot \alpha_+$, we transition to $(k+1, z)$, if it is in Ω.
- With the remaining probability, we just stay at (k, z).

Consider the distribution

$$\nu(k, z) \propto \left(\frac{\alpha_+}{\alpha_-}\right)^k.$$

It is easy to see that this Markov chain satisfies detailed balance with ν as its stationary distribution. In particular,

$$\begin{aligned}
\nu(k, z) \cdot T((k-1, z)|(k, z)) &= \left(\frac{\alpha_+}{\alpha_-}\right)^k \cdot \frac{1}{4} \cdot \alpha_- \\
&= \left(\frac{\alpha_+}{\alpha_-}\right)^{k-1} \cdot \frac{1}{4} \cdot \alpha_+ \\
&= \nu(k-1, z) \cdot T((k, z)|(k-1, z)).
\end{aligned}$$

So ν will be a stationary distribution of the minibatch MH chain $\hat{T}$.

Observe that the marginal distribution of k in π is

$$\pi(k) = \int_0^{\exp(kX)} \pi(k, z)\, dz \propto \exp\left(-\sum_{i=1}^{N} k \cdot x_i\right) \cdot \exp(kX) = 1,$$

so the marginal distribution of k in the target distribution is actually the uniform distribution. On the other hand, using the same derivation, the marginal

distribution of k in ν is

$$\nu(k) \propto \left(\frac{\alpha_+}{\alpha_-}\right)^k \cdot \exp(kX) = \left(\frac{\alpha_+}{\alpha_-} \cdot \exp(X)\right)^k.$$

We know immediately by substituting our definitions of α_+ and α_- into (D.1) that

$$\frac{\alpha_-}{\alpha_+} \neq \frac{\mu(\theta')}{\mu(\theta)} = \exp\left(\sum_{i=1}^{N}(V_i(\theta) - V_i(\theta')\right) = \exp\left(\sum_{i=1}^{N} x_i\right) = \exp(X).$$

As a consequence, we know that

$$\frac{\alpha_+}{\alpha_-} \cdot \exp(X) \neq 1.$$

Call this constant

$$A = \frac{\alpha_+}{\alpha_-} \cdot \exp(X),$$

and observe that $A \neq 1$ and that A is independent of our choice of K (which still remains unset). This gives

$$\nu(k) \propto A^k.$$

Explicitly, this distribution will be

$$\nu(k) = \frac{1}{\sum_{k=0}^{K-1} A^k} \cdot A^k = \frac{1 - A}{1 - A^K} \cdot A^k.$$

Since the total variation distance between two probability measures is lower bounded by the TV-distance between their marginal distributions in any one variable, and similarly the KL divergence is *also* lower bounded by the KL divergence between its marginal distributions in any one variable (both these facts follow directly from the monotonicity property of the f-divergence, of which the KL-divergence and TV-distance are both instances), to prove this theorem it suffices to show both TV-distance and KL-divergence bounds on the marginal distributions in k. We do this now.

Bounding the total variation distance. Now, we compute the total variation distance between π and ν. For this bit of the proof, we will just consider the marginal distribution in k, as this provides a lower bound on the TV distance between the joint distribution. For simplicity, for the rest of the proof, we let $\tilde{\pi}$ denote this marginal distribution of k in ν, and also let π denote the marginal distribution of k in π. By the definition of total variation distance,

$$\begin{aligned}\mathrm{TV}(\pi, \tilde{\pi}) &= \frac{1}{2} \sum_{k=0}^{K-1} |\tilde{\pi}(k) - \pi(k)| \\ &= \frac{1}{2} \sum_{k=0}^{K-1} \left| \frac{1-A}{1-A^K} \cdot A^k - \frac{1}{K} \right|.\end{aligned}$$

If $A < 1$,

$$\begin{aligned}\mathrm{TV}(\pi, \tilde{\pi}) &= \sum_{k=0}^{K_0} \left(\frac{1-A}{1-A^K} \cdot A^k - \frac{1}{K} \right) \\ &= \frac{1-A^{K_0}}{1-A^K} - \frac{K_0}{K} \end{aligned} \tag{D.2}$$

where K_0 is the largest k such that

$$\frac{1-A}{1-A^K} \cdot A^k > \frac{1}{K}.$$

By solving the above equation, we have

$$K_0 = \left\lfloor \frac{\log\left(1 - A^K\right) - \log(1-A) - \log(K)}{\log(A)} \right\rfloor.$$

We can lower bound K_0 by

$$\begin{aligned}K_0 &\geq \frac{\log\left(1 - A^K\right) - \log(1-A) - \log(K)}{\log(A)} - 1 \\ &\geq \frac{-\log(1-A) - \log(K)}{\log(A)} - 1.\end{aligned}$$

It follows that the first term in (D.2) becomes

$$\frac{1-A^{K_0}}{1-A^K} \geq \frac{1 - \frac{1}{KA(1-A)}}{1-A^K} \geq 1 - \frac{1}{KA(1-A)}.$$

We can also upper bound K_0 and then the second term can be bounded as the following

$$\frac{K_0}{K} \leq \frac{\log\left(1 - A^K\right) - \log(K)}{K \log(A)}.$$

When $K \geq \frac{\log\left(1-\exp\left(-\frac{1}{2}\right)\right)}{\log(A)}$, we have $\log\left(1 - A^K\right) \geq -\frac{1}{2}$. Since $\log(K) \leq K^{\frac{1}{2}}$ and $K^{-1} \leq K^{-\frac{1}{2}}$, we have

$$\frac{K_0}{K} \leq \frac{-\frac{1}{2}K^{-1} - K^{-\frac{1}{2}}}{\log(A)} \leq -\left(\frac{3}{2\log(A)}\right)K^{-\frac{1}{2}}.$$

Therefore, the TV distance is bounded by

$$\begin{aligned}\mathrm{TV}(\pi, \tilde{\pi}) &\geq 1 - \frac{1}{KA(1-A)} + \left(\frac{3}{2\log(A)}\right)K^{-\frac{1}{2}} \\ &\geq 1 + \left(\frac{3}{2\log(A)} - \frac{1}{A(1-A)}\right)K^{-\frac{1}{2}}.\end{aligned}$$

To make $\mathrm{TV}(\pi, \tilde{\pi}) \geq \delta$, we just need to set

$$K \geq \frac{\left(\frac{3}{2\log(A)} - \frac{1}{A(1-A)}\right)^2}{(1-\delta)^2}.$$

Similarly, if $A > 1$,

$$\begin{aligned}\mathrm{TV}(\pi, \tilde{\pi}) &= \sum_{k=K_0}^{K-1}\left(\frac{1-A}{1-A^K} \cdot A^k - \frac{1}{K}\right) \\ &= \frac{A^K - A^{K_0}}{A^K - 1} - \frac{K - K_0}{K} \\ &= \frac{K_0}{K} - \frac{A^{K_0} - 1}{A^K - 1}\end{aligned}$$

where

$$K_0 = \left\lceil \frac{\log\left(A^K - 1\right) - \log(A-1) - \log(K)}{\log(A)} \right\rceil$$

which is the smallest k such that

$$\frac{1-A}{1-A^K} \cdot A^k > \frac{1}{K}.$$

We can get an upper bound of K_0 by

$$\begin{aligned} K_0 &\leq \frac{\log\left(A^K - 1\right) - \log(A-1) - \log(K)}{\log(A)} + 1 \\ &= \log_A\left(\frac{A^K - 1}{K(A-1)}\right) + 1. \end{aligned}$$

Therefore,

$$\begin{aligned} \frac{A^{K_0} - 1}{A^K - 1} &\leq \frac{A \cdot \left(\frac{A^K - 1}{K(A-1)}\right) - 1}{A^K - 1} \\ &= \frac{A}{K(A-1)} - \frac{1}{A^K - 1}. \end{aligned}$$

We can lower bound K_0 by

$$K_0 \geq \log_A\left(A^K - 1\right) - \log_A(A-1) - \log_A(K).$$

When $K \geq 1 - \log_A(A-1)$, $A^K - 1 \geq A^{K-1}$. Then we have

$$\begin{aligned} K_0 &\geq \log_A\left(A^{K-1}\right) - \log_A(A-1) - \log_A(K) \\ &= K - 1 - \log_A(A-1) - \log_A(K). \end{aligned}$$

It follows that

$$\frac{K_0}{K} \geq 1 - \frac{1}{K} - \frac{\log_A(A-1)}{K} - \frac{\log_A(K)}{K}.$$

Since $\log(K) \leq K^{\frac{1}{2}}$ and $K^{-1} \leq K^{-\frac{1}{2}}$, the TV distance can be bounded by

$$\begin{aligned} \mathrm{TV}(\pi, \tilde{\pi}) &\geq 1 - \frac{1}{K} - \frac{\log_A(A-1)}{K} - \frac{\log_A(K)}{K} - \frac{A}{K(A-1)} + \frac{1}{A^K - 1} \\ &\geq 1 - \left(1 + \log_A(A-1) + \frac{1}{\log(A)} + \frac{A}{A-1}\right) K^{-\frac{1}{2}}. \end{aligned}$$

To make $\mathrm{TV}(\pi, \tilde{\pi}) \geq \delta$, we just need

$$K \geq \left(\frac{1 + \log_A(A-1) + \frac{1}{\log(A)} + \frac{A}{A-1}}{1 - \delta}\right)^2.$$

Since we could set K arbitrarily, it is clear that we can do this.

Bounding the KL divergence. We can compute KL divergence between π and $\tilde{\pi}$ as follows

$$\begin{aligned}
\mathrm{KL}(\pi, \tilde{\pi}) &= \sum_{k=0}^{K-1} \frac{1}{K} \cdot \log\left(\frac{1}{K} \cdot \frac{1-A^K}{(1-A)A^k}\right) \\
&= \frac{1}{K} \cdot \sum_{k=0}^{K-1} \left[\log\left(\frac{1}{K} \cdot \frac{1-A^K}{(1-A)}\right) - k\log(A)\right] \\
&= \log\left(\frac{1-A^K}{K(1-A)}\right) - \frac{\log(A)}{K} \sum_{k=0}^{K-1} k \\
&= \log\left(\frac{1-A^K}{K(1-A)}\right) - \frac{(K-1)\log(A)}{2}
\end{aligned}$$

If $A < 1$, we have

$$\begin{aligned}
\mathrm{KL}(\pi, \tilde{\pi}) &= \log\left(1-A^K\right) - \log((1-A)K) - \frac{K\log(A)}{2} + \frac{\log(A)}{2} \\
&\geq \log\left(1-A^K\right) - \left(\frac{1-A+\log(A)}{2}\right)K + \frac{\log(A)}{2}.
\end{aligned}$$

The last equation is because $\log(x) \leq \frac{x}{2}$.

To further simplify the above equation, we first note that $1 - A + \log(A) < 0$ when $A \neq 1$. And then when $K \geq \log_A\left(1 - A^{\frac{1}{2}}\right)$, we have $1 - A^K \geq A^{\frac{1}{2}}$. It follows that we can simplify it to be

$$\mathrm{KL}(\pi, \tilde{\pi}) \geq \log(A) - \left(\frac{1-A+\log(A)}{2}\right)K.$$

To make $\mathrm{KL}(\pi, \tilde{\pi}) \geq \rho$, it is clear that we just need to set

$$K \geq \frac{2(\rho - \log(A))}{A - 1 - \log(A)}.$$

Consider when $A > 1$,

$$\mathrm{KL}(\pi, \tilde{\pi}) = \log\left(\frac{A^K - 1}{K(A-1)}\right) - \frac{(K-1)\log(A)}{2}.$$

If $K \geq \frac{\log(2)}{\log(A)}$, we have that $A^K - 1 \geq \frac{A^K}{2}$. It follows that

$$\begin{aligned}\mathrm{KL}(\pi, \tilde{\pi}) &\geq K\log(A) - \log(K) - \log(2A-2) - \frac{K\log(A)}{2} \\ &= \frac{K\log(A)}{2} - \log(K) - \log(2A-2).\end{aligned}$$

To make $\mathrm{KL}(\pi, \tilde{\pi}) \geq \rho$, we need

$$\frac{K\log(A)}{2} - \log(K) \geq \rho + \log(2A-2).$$

Let $K = \exp(y)$. By Taylor series, we know $\exp(y) \geq \frac{y^2}{2}$. Then it follows that

$$\frac{y^2\log(A)}{4} - y \geq \rho + \log(2A-2).$$

Solve the above inequality, we can get

$$y \geq \frac{1 + 2 \cdot \frac{\log(A)}{4} \cdot \Big(\rho + \log(2A-2)\Big)}{2 \cdot \frac{\log(A)}{4}} = \frac{2 + \log(A)\Big(\rho + \log(2A-2)\Big)}{\log(A)}.$$

It follows that it suffices to set

$$K \geq \exp\left(\frac{2 + \log(A)\Big(\rho + \log(2A-2)\Big)}{\log(A)}\right).$$

Concluding the proof. The theorem now follows from choosing a K large enough that both the TV distance inequality we derived and the KL divergence inequality we derived are satisfied. □

D.2 Connection between Theorem 2 and TV Bound of Inexact MH Methods

Some inexact methods such as MHSubLhd [44] have bounded TV distance between the target distribution and the approximate distribution (see Proposition 3.2 in Bardenet et al. [44]). We would like to emphasize that Theorem 2

is compatible with these results. Specifically, Proposition 3.2 assumes P_{MH} has a bounded mixing time. It is well known that this produces a TV bound for any kernel by coupling [365]. Our theorem does not have this assumption; it suggests that for MHSubLhd, with a given user-specified error, there exists a target distribution and proposal satisfying Theorem 2, on which P_{MH} either does not have bounded mixing time or the mixing time is large enough such that the TV bound is greater than δ.

D.3 Proof of Statement 1

Proof. We prove this by construction. Consider a dataset $\{x_i\}_{i=1}^N$. The data instances can take two values $\{-\frac{M}{N}, \frac{M}{N}\}$ where M is a positive constant. Assume that half of the data instances take value $\frac{M}{N}$ and the remaining take $-\frac{M}{N}$. Let the target distribution be $\pi(\theta) = \frac{1}{Z}\exp\left(\theta \cdot \sum_{i=1}^N x_i\right)$ and the domain for θ be $\{0, 1, \ldots, K-1\}$. We define the proposal distribution to be the following

$$p(\theta, \theta) = \frac{1}{2}, \quad \text{for all } \theta; \quad p(\theta, \theta-1) = \frac{1}{4}, \quad p(\theta, \theta+1) = \frac{1}{4} \quad \text{for } \theta \in \{1, \ldots, K-2\};$$

and $p(0,1) = p(K-1, K-2) = \frac{1}{2}$.

Recall that FMH factorizes the target distribution $\pi(\theta)$ and the proposal distribution $p(\theta)$ as follows

$$\pi(\theta) \propto \prod_{i=1}^m \pi_i(\theta), \qquad p(\theta) \propto \prod_{i=1}^m p_i(\theta)$$

where $m \geq 1$ and π_i and p_i are some non-negative functions. Then the acceptance rate is given by[1]

$$a_{\text{FMH}}(\theta, \theta') = \prod_{i=1}^m \min\left(1, \frac{\pi(\theta')p_i(\theta, \theta')}{\pi(\theta)p_i(\theta', \theta)}\right).$$

[1]There are typos with prime (′) marks in this expression in the camera-ready version of the paper.

A common choice is to set $m = N$. On this example, we can write the acceptance rate of transitioning from θ to $\theta' = \theta + 1$ in FMH as follows

$$a_{\text{FMH}}(\theta, \theta') = \prod_{i=1}^{N} \min\left(1, \exp(x_i)\right) = \left(\exp\left(-\frac{M}{N}\right)\right)^{\frac{N}{2}} = \exp\left(-\frac{M}{2}\right).$$

It is easy to show that the acceptance rate of transitioning from θ to $\theta' = \theta - 1$ in FMH is the same.

When $M > -2\log(p)$, it is clear that the acceptance rate of FMH is less than p. By contrast, the acceptance rate of standard MH is

$$a_{\text{MH}}(\theta, \theta') = \min\left(1, \exp\left(\pm \sum_{i=1}^{N} x_i\right)\right) = 1.$$

In order to preserve geometric ergodicity, Cornish et al. [155] introduces *truncated FMH* (TFMH) which forces FMH degrade to standard MH when the energy exceeds a threshold R. If we set hyperparameter $R > M/2$, then in each step, the value of a_{TFMH} will be the same as a_{FMH}. Therefore, if setting $M > -2\log(p)$, we have

$$\frac{a_{\text{TFMH}}}{a_{\text{MH}}} \leq \frac{p}{1} = p.$$

If we set $R \leq M/2$, TFMH falls back to standard, full-batch MH — using the whole dataset at each step. This proves the statement. □

D.4 Construction of Algorithm 4

Algorithm 4 can be derived by carefully replacing the global bounds on the energy in PoissonMH [643] with local bounds on the energy differences (Assumption 1).[2] PoissonMH is a variant of Poisson Gibbs and therefore inherits

[2] In our construction of Algorithm 9, we similarly have typos in the camera-ready paper around misplaced ′ marks in the acceptance ratio.

Algorithm 9 PoissonMH

given: initial state $\theta \in \Theta$; proposal dist. q; hyperparameter λ; Global bounds M_i, L
loop
  propose $\theta' \sim q(\cdot|\theta)$
  for $i \in \{1, \ldots, N\}$ **do**
    sample $s_i \sim \text{Poisson}\left(\frac{\lambda M_i}{L} + \phi_i(\theta)\right)$
  end for
  form minibatch $\mathcal{S} \leftarrow \{i | s_i > 0\}$
  compute MH ratio $r \leftarrow \frac{\exp\left(\sum_{i\in\mathcal{S}} s_i \log\left(1+\frac{L}{\lambda M_i}\phi_i(\theta)\right)\right) q(\theta|\theta')}{\exp\left(\sum_{i\in\mathcal{S}} s_i \log\left(1+\frac{L}{\lambda M_i}\phi_i(\theta')\right)\right) q(\theta'|\theta)}$
  with probability $\min(1, r)$, set $\theta \leftarrow \theta'$
end loop

the same assumptions for Gibbs sampling on graphical models, which are often violated in the applications of MH. In particular, PoissonMH works on *factor graphs* which define a distribution $\pi(\theta)$ over a set of factors $\{\phi_i(\theta)\}_{i=1}^N$ as follows

$$\pi(\theta) \propto \exp\left(\sum_{i=1}^{N} \phi_i(\theta)\right).$$

PoissonMH assumes that each factor ϕ_i is non-negative without the loss of generality (we can add a positive constant to ϕ_i to make it non-negative without changing the distribution) and is bounded globally by a constant M_i. That is

$$0 \leq \phi_i(\theta) \leq M_i \text{ for all } \theta.$$

This assumption does not hold for most applications of MH, such as the linear and logistic regression experiments in Section 5.5.

Let $L = \sum_i M_i$ and define Poisson auxiliary variable s_i as the following

$$s_i|\theta \sim \text{Poisson}\left(\frac{\lambda M_i}{L} + \phi_i(\theta)\right),$$

where $\lambda > 0$ is a hyperparameter. Running standard MH on the joint distribu-

tion of θ and s_i results in the following acceptance ratio[3]

$$_{\text{PoissonMH}}(\theta, \theta') = \frac{\exp\left(\sum_i s_i \log\left(1 + \frac{L}{\lambda M_i}\phi_i(\theta)\right)\right) q(\theta|\theta')}{\exp\left(\sum_i s_i \log\left(1 + \frac{L}{\lambda M_i}\phi_i(\theta')\right)\right) q(\theta'|\theta)}.$$

Here, the sum is essentially performed over the set of index i whose s_i is greater than zero. When $s_i = 0$, it is clear that the factor ϕ_i will not appear in the acceptance ratio $r_{\text{PoissonMH}}$. Thus PoissonMH enables using a subset of factors for the MH decision step (Algorithm 9).

To construct our method from this, we can define the factor ϕ_i in the factor graph to be

$$\phi_i(x) = \frac{U_i(\theta) + U_i(\theta')}{2} - U_i(x) + \frac{c_i}{2} M(\theta, \theta') \tag{D.3}$$

where $x \in \{\theta, \theta'\}$. It is easy to see that ϕ_i satisfy $0 \leq \phi_i(x) \leq c_i M(\theta, \theta')$. And then we define the Poisson variables s_i as the follows

$$s_i|(\theta, \theta') \sim \text{Poisson}\left(\frac{\lambda c_i}{C} + \phi_i(\theta)\right) = \text{Poisson}\left(\frac{\lambda c_i}{C} + \frac{U_i(\theta') - U_i(\theta) + c_i M(\theta, \theta')}{2}\right).$$

These Poisson auxiliary variables $\{s_i\}_{i=1}^N$ are called *local*, because their distributions change each iteration depending on the current pair (θ, θ') and only rely on local bounds in Assumption 1. This is in contrast to the *global* auxiliary variables used in PoissonMH and FlyMC which are used to form a joint distribution with θ and both require global bounds in their conditional distributions.

The acceptance ratio r_{TunaMH} is the same as $r_{\text{PoissonMH}}$ but with the new definitions of s_i and ϕ_i. We outline TunaMH using the notation of ϕ_i and s_i in Algorithm 10.

[3]The misplaced ′ marks in the acceptance test also carry through here, and are edited / differ from the current camera-ready version.

We now show that Algorithm 10 is statistically equivalent to Algorithm 4. To see this, we first use *thinning*, a commonly used technique [61, 77, 155, 368, 643], to quickly resample all s_i from their new distributions in each iteration in Algorithm 10. This is achieved by replacing the global bounds with the local bounds in Algorithm 4 in the Appendix of Zhang and De Sa [643]. Specifically, we first sample B from a Poisson distribution

$$B \sim \text{Poisson}(\lambda + CM(\theta, \theta')).$$

Here $\lambda + CM(\theta, \theta')$ is an upper bound on $\mathbf{E}[\sum_i s_i]$. We then form the minibatch by running

for $b \in \{1, \ldots, B\}$ **do**
  sample i_b such that $\mathbf{P}(i_b = i) = c_i/C$, for $i = 1 \ldots N$
  with probability $\frac{\lambda c_{i_b} + C\phi_{i_b}(\theta)}{\lambda c_{i_b} + Cc_{i_b}M(\theta,\theta')}$ **add** i_b to $\mathcal{I}$
end for

By substituting $\lambda = \chi C^2 M^2(\theta, \theta')$ and the expression of ϕ_i, we can get the part of "form minibatch $\mathcal{I}$" in Algorithm 4.

To see that the MH ratio in Algorithm 4 and 10 are equivalent, we can write out r in Algorithm 10[4] using the above fast way of resampling s_i

$$r_{\text{TunaMH}} = \frac{\exp\left(\sum_{i\in\mathcal{I}} \log\left(1 + \frac{C}{\lambda c_i}\phi_i(\theta)\right)\right) q(\theta|\theta')}{\exp\left(\sum_{i\in\mathcal{I}} \log\left(1 + \frac{C}{\lambda c_i}\phi_i(\theta')\right)\right) q(\theta'|\theta)}.$$

We then substitute the definition of ϕ_i in (D.3) and it follows that

$$r_{\text{TunaMH}} = \exp\left(\sum_{i\in\mathcal{I}} \left(\log\left(\frac{2\lambda c_i + C\left(U_i(\theta) - U_i(\theta') + c_i M(\theta, \theta')\right)}{2\lambda c_i + C\left(U_i(\theta') - U_i(\theta) + c_i M(\theta, \theta')\right)}\right)\right)\right) \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}.$$

[4] Again, the $'$ marks in the camera-ready version of this paper are flipped in the acceptance test, and corrected here.

Algorithm 10 TunaMH

given: initial state $\theta \in \Theta$; proposal dist. q; λ; Asm. 1 parameters c_i, C, M; function ϕ_i defined in (D.3)
loop
  propose $\theta' \sim q(\cdot|\theta)$ and **compute** $M(\theta, \theta')$
  for $i \in \{1, \ldots, N\}$ **do**
    sample $s_i \sim \text{Poisson}\left(\frac{\lambda c_i}{C} + \phi_i(\theta)\right)$
  end for
  form minibatch $\mathcal{S} \leftarrow \{i | s_i > 0\}$
  compute MH ratio $r \leftarrow \frac{\exp\left(\sum_{i\in\mathcal{S}} s_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)\right) q(\theta|\theta')}{\exp\left(\sum_{i\in\mathcal{S}} s_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)\right) q(\theta'|\theta)}$
  with probability $\min(1, r)$, set $\theta \leftarrow \theta'$
end loop

We can rearrange the log term inside r_{TunaMH} as

$$\begin{aligned}
&\log\left(\frac{2\lambda c_i + C\left(U_i(\theta) - U_i(\theta') + c_i M(\theta,\theta')\right)}{2\lambda c_i + C\left(U_i(\theta') - U_i(\theta) + c_i M(\theta,\theta')\right)}\right) \\
&\quad= \log\left(\frac{2\lambda c_i + C\left(U_i(\theta) - U_i(\theta')\right) + c_i C M(\theta,\theta')}{2\lambda c_i + C\left(U_i(\theta') - U_i(\theta)\right) + c_i C M(\theta,\theta')}\right) \\
&\quad= \log\left(\frac{1 + \frac{C}{2\lambda c_i + c_i C M(\theta,\theta')}\left(U_i(\theta) - U_i(\theta')\right)}{1 + \frac{C}{2\lambda c_i + c_i C M(\theta,\theta')}\left(U_i(\theta') - U_i(\theta)\right)}\right) \\
&\quad= 2\,\text{artanh}\left(\frac{C\left(U_i(\theta) - U_i(\theta')\right)}{c_i(2\lambda + C M(\theta,\theta'))}\right).
\end{aligned}$$

So r_{TunaMH} can be written as[5]

$$r_{\text{TunaMH}} = \exp\left(2\sum_{i\in\mathcal{I}} \text{artanh}\left(\frac{C\left(U_i(\theta) - U_i(\theta')\right)}{c_i(2\lambda + C M(\theta,\theta'))}\right)\right) \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}.$$

Finally setting λ to be $\chi C^2 M^2(\theta, \theta')$ produces the MH ratio in Algorithm 4.

By proving the equivalence of the minibatch and the MH ratio, we show that Algorithm 4 and 10 are statistically equivalent.

[5] Again, fixed $'$ marks in the proposal ratio.

D.5 Proof of Theorem 3

In this section, we prove Theorem 3, which asserts that TunaMH is reversible and has stationary distribution π, and gives bounds on its spectral gap relative to the spectral gap of the original Metropolis-Hastings algorithm.

Proof. For convenience, we prove Theorem 3 using Algorithm 10 statement which is statistically equivalent to Algorithm 4. The transition operator can be written as the following

$$
\begin{aligned}
&T(\theta, \theta') \\
&= \mathbf{E}\left\{ q(\theta'|\theta) \min\left(1, \frac{q(\theta|\theta') \exp\left(\sum_i \left[s_i \log\left(\frac{\lambda c_i}{C} + \phi_i(\theta')\right) - \log s_i! \right]\right)}{q(\theta'|\theta) \exp\left(\sum_i \left[s_i \log\left(\frac{\lambda c_i}{C} + \phi_i(\theta)\right) - \log s_i! \right]\right)}\right)\right\} \\
&= \mathbf{E}\left\{ q(\theta'|\theta) \min\left(1, \frac{q(\theta|\theta') \exp\left(\sum_i \left[s_i \log\left(\frac{\lambda c_i}{C} + \phi_i(\theta')\right)\right]\right)}{q(\theta'|\theta) \exp\left(\sum_i \left[s_i \log\left(\frac{\lambda c_i}{C} + \phi_i(\theta)\right)\right]\right)}\right)\right\} \\
&= \sum_s \left\{ q(\theta'|\theta) \min\left(1, \frac{q(\theta|\theta') \exp\left(\sum_i \left[s_i \log\left(\frac{\lambda c_i}{C} + \phi_i(\theta')\right)\right]\right)}{q(\theta'|\theta) \exp\left(\sum_i \left[s_i \log\left(\frac{\lambda c_i}{C} + \phi_i(\theta)\right)\right]\right)}\right)\right\} \prod_i p(s_i|\theta, \theta') \\
&= \sum_s \left\{ q(\theta'|\theta) \min\left(\exp\left(\sum_i \left[s_i \log\left(\frac{\lambda c_i}{C} + \phi_i(\theta)\right) - \phi_i(\theta) - \frac{\lambda c_i}{C} - \log s_i! \right]\right), \right.\right. \\
&\qquad \left.\left. \frac{q(\theta|\theta') \exp\left(\sum_i \left[s_i \log\left(\frac{\lambda c_i}{C} + \phi_i(\theta')\right)\right]\right)}{q(\theta'|\theta) \exp\left(\sum_i \phi_i(\theta) + \frac{\lambda c_i}{C} + \log s_i!\right)}\right)\right\} \\
&= \sum_s \left\{ q(\theta'|\theta) \min\left(\exp\left(\sum_i \left[s_i \log\left(\frac{\lambda c_i}{C} + \phi_i(\theta)\right) - \phi_i(\theta) - \frac{\lambda c_i}{C} - \log s_i! \right]\right), \right.\right. \\
&\qquad \left.\left. \frac{q(\theta|\theta')}{q(\theta'|\theta)} \exp\left(\sum_i \left[s_i \log\left(\frac{\lambda c_i}{C} + \phi_i(\theta')\right) - \phi_i(\theta) - \frac{\lambda c_i}{C} - \log s_i! \right]\right)\right)\right\}
\end{aligned}
$$

Multiplying $\pi(\theta)$ to both sides produces

$$
\begin{aligned}
&\pi(\theta)T(\theta,\theta') \\
&\quad= \frac{1}{Z}\exp\left(-\sum_i U_i(\theta)\right)T(\theta,\theta') \\
&\quad= \frac{1}{Z}\sum_s \min\Bigg(q(\theta'|\theta)\exp\Bigg(\sum_i\Bigg[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta)\right) \\
&\qquad\qquad -\frac{U_i(\theta)+U_i(\theta')}{2}-\frac{c_i}{2}M(\theta,\theta')-\frac{\lambda c_i}{C}-\log s_i!\Bigg]\Bigg), \\
&\qquad q(\theta|\theta')\exp\Bigg(\sum_i\Bigg[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta')\right) \\
&\qquad\qquad -\frac{U_i(\theta)+U_i(\theta')}{2}-\frac{c_i}{2}M(\theta,\theta')-\frac{\lambda c_i}{C}-\log s_i!\Bigg]\Bigg)\Bigg)\Bigg).
\end{aligned}
$$

It is clear that the expression is symmetric in θ and θ'. Therefore the chain is reversible and its stationary distribution is $\pi(\theta)$. This proves the first part of the theorem.

To prove the second part of the theorem, the bound on the spectral gap, we continue to reduce the transition probability in the previous proof to

$$
\begin{aligned}
&\pi(\theta)T(\theta,\theta') \\
&\quad= \frac{1}{Z}\sum_s \min\Bigg(q(\theta'|\theta)\exp\Big(\sum_i\Big[s_i\log\Big(\frac{\lambda c_i}{C}+\phi_i(\theta)\Big) \\
&\qquad\quad -\frac{U_i(\theta)+U_i(\theta')}{2}-\frac{c_i}{2}M(\theta,\theta')-s_i\log\frac{\lambda c_i}{C}\Big]\Big), \\
&\qquad q(\theta|\theta')\exp\Big(\sum_i\Big[s_i\log\Big(\frac{\lambda c_i}{C}+\phi_i(\theta')\Big) \\
&\qquad\quad -\frac{U_i(\theta)+U_i(\theta')}{2}-\frac{c_i}{2}M(\theta,\theta')-s_i\log\frac{\lambda c_i}{C}\Big]\Big)\Bigg) \\
&\qquad\cdot\prod_i\frac{1}{s_i!}\exp\Big(-\frac{\lambda c_i}{C}\Big)\Big(\frac{\lambda c_i}{C}\Big)^{s_i} \\
&\quad= \frac{1}{Z}\sum_s \min\Bigg(q(\theta'|\theta)\exp\Big(\sum_i\Big[s_i\log\Big(1+\frac{C}{\lambda c_i}\phi_i(\theta)\Big) \\
&\qquad\quad -\frac{U_i(\theta)+U_i(\theta')}{2}-\frac{c_i}{2}M(\theta,\theta')\Big]\Big), \\
&\qquad q(\theta|\theta')\exp\Big(\sum_i\Big[s_i\log\Big(1+\frac{C}{\lambda c_i}\phi_i(\theta')\Big)-\frac{U_i(\theta)+U_i(\theta')}{2}-\frac{c_i}{2}M(\theta,\theta')\Big]\Big)\Bigg) \\
&\qquad\cdot\prod_i\frac{1}{s_i!}\exp\Big(-\frac{\lambda c_i}{C}\Big)\Big(\frac{\lambda c_i}{C}\Big)^{s_i}.
\end{aligned}
$$

Note that s_i here are non-negative integers that a Poisson variable can take, not variables. So if we let $r_i \sim \text{Poisson}\left(\frac{\lambda c_i}{C}\right)$ and r_i to be all independent, we can write this as

$$
\begin{aligned}
\pi(\theta)T(\theta,\theta') = \frac{1}{Z}\mathbf{E}\min\Bigg(&q(\theta'|\theta)\exp\Big(\sum_i r_i\log\Big(1+\frac{C}{\lambda c_i}\phi_i(\theta)\Big)\Big), \\
&q(\theta|\theta')\exp\Big(\sum_i r_i\log\Big(1+\frac{C}{\lambda c_i}\phi_i(\theta')\Big)\Big)\Bigg) \\
&\cdot\exp\Big[-\frac{1}{2}\Big(\sum_i U_i(\theta)+\sum_i U_i(\theta')+CM(\theta,\theta')\Big)\Big].
\end{aligned}
$$

Assume $G(\theta, \theta')$ is the transition operator of standard MH. Consider the ratio

$$
\begin{aligned}
&\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} \\
&= \frac{1}{Z}\mathbf{E}\min\left(q(\theta'|\theta)\exp\left(\sum_i r_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)\right),\right. \\
&\qquad \left. q(\theta|\theta')\exp\left(\sum_i r_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)\right)\right) \\
&\qquad \cdot \exp\left[-\frac{1}{2}\left(\sum_i U_i(\theta)+\sum_i U_i(\theta')+CM(\theta,\theta')\right)\right] \\
&\qquad \cdot \left[1\Big/\left(\frac{1}{Z}\min\left(q(\theta'|\theta)\exp\left(-\sum_i U_i(\theta)\right), q(\theta|\theta')\exp\left(-\sum_i U_i(\theta')\right)\right)\right)\right].
\end{aligned}
$$

We know that $\frac{\min(A,B)}{\min(C,D)} = \min\left(\frac{A}{\min(C,D)}, \frac{B}{\min(C,D)}\right) \geq \min\left(\frac{A}{C}, \frac{B}{D}\right)$. The last inequality is due to the fact that $\frac{1}{\min(C,D)} \geq \frac{1}{C}$ and $\frac{1}{\min(C,D)} \geq \frac{1}{D}$.

With this inequality, we can continue simplifying the ratio,

$$
\begin{aligned}
&\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} \\
&\quad \geq \mathbf{E}\left[\min\left(\frac{\exp\left(\sum_i r_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)\right)}{\exp\left(-\sum_i U_i(\theta)\right)}, \frac{\exp\left(\sum_i r_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)\right)}{\exp\left(-\sum_i U_i(\theta')\right)}\right)\right] \\
&\qquad \cdot \exp\left[-\frac{1}{2}\left(\sum_i U_i(\theta)+\sum_i U_i(\theta')+CM(\theta,\theta')\right)\right] \\
&\quad = \mathbf{E}\left[\min\left(\exp\left(\sum_i\left(r_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)-\phi_i(\theta)\right)\right),\right.\right. \\
&\qquad \left.\left. \exp\left(\sum_i\left(r_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)-\phi_i(\theta')\right)\right)\right)\right] \\
&\quad = \mathbf{E}\left[\max\left(\exp\left(\sum_i\left(\phi_i(\theta)-r_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)\right)\right),\right.\right. \\
&\qquad \left.\left. \exp\left(\sum_i\left(\phi_i(\theta')-r_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)\right)\right)\right)^{-1}\right].
\end{aligned}
$$

Because $f(x) = \frac{1}{x}$ is a convex function, by Jensen's inequality it follows

$$\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} \geq \mathbf{E}\Bigg[\max\Bigg(\exp\Bigg(\sum_i \Bigg(\phi_i(\theta) - r_i \log\Bigg(1 + \frac{C}{\lambda c_i}\phi_i(\theta)\Bigg)\Bigg)\Bigg), \exp\Bigg(\sum_i \Bigg(\phi_i(\theta') - r_i \log\Bigg(1 + \frac{C}{\lambda c_i}\phi_i(\theta')\Bigg)\Bigg)\Bigg)\Bigg)\Bigg]^{-1}.$$

We use $\max(A, B) \leq (A^p + B^p)^{\frac{1}{p}}$ to remove the max function.

$$\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} \geq \mathbf{E}\Bigg[\Bigg(\exp\Bigg(p \sum_i \Bigg(\phi_i(\theta) - r_i \log\Bigg(1 + \frac{C}{\lambda c_i}\phi_i(\theta)\Bigg)\Bigg)\Bigg) + \exp\Bigg(p \sum_i \Bigg(\phi_i(\theta') - r_i \log\Bigg(1 + \frac{C}{\lambda c_i}\phi_i(\theta')\Bigg)\Bigg)\Bigg)\Bigg)^{\frac{1}{p}}\Bigg]^{-1}.$$

Since $x^{\frac{1}{p}}$ is concave, by Jensen's inequality

$$\begin{aligned}\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} &\geq \mathbf{E}\Bigg[\exp\Bigg(p \sum_i \Bigg(\phi_i(\theta) - r_i \log\Bigg(1 + \frac{C}{\lambda c_i}\phi_i(\theta)\Bigg)\Bigg)\Bigg) + \\ &\qquad \exp\Bigg(p \sum_i \Bigg(\phi_i(\theta') - r_i \log\Bigg(1 + \frac{C}{\lambda c_i}\phi_i(\theta')\Bigg)\Bigg)\Bigg)\Bigg]^{-\frac{1}{p}} \\ &= \Bigg[\prod_i \mathbf{E} \exp\Bigg(p\phi_i(\theta) - p r_i \log\Bigg(1 + \frac{C}{\lambda c_i}\phi_i(\theta)\Bigg)\Bigg) + \\ &\qquad \prod_i \mathbf{E} \exp\Bigg(p\phi_i(\theta') - p r_i \log\Bigg(1 + \frac{C}{\lambda c_i}\phi_i(\theta')\Bigg)\Bigg)\Bigg]^{-\frac{1}{p}}.\end{aligned}$$

$\mathbf{E}\Big[\exp\Big(- p r_i \log\big(1 + \frac{C}{\lambda c_i}\phi_i(\theta)\big)\Big)\Big]$ is the moment generating function of the Poisson random variable r_i evaluated at

$$t = -p \log\Bigg(1 + \frac{C}{\lambda c_i}\phi_i(\theta)\Bigg).$$

We know that

$$\mathbf{E} \exp(r_i t) = \exp\Big(\frac{\lambda c_i}{C}(\exp(t) - 1)\Big),$$

therefore,

$$\mathbf{E}\Bigg[\exp\Bigg(- p r_i \log\Bigg(1 + \frac{C}{\lambda c_i}\phi_i(\theta)\Bigg)\Bigg)\Bigg] = \exp\Bigg(\frac{\lambda c_i}{C}\Bigg(1 + \frac{C}{\lambda c_i}\phi_i(\theta)\Bigg)^{-p} - \frac{\lambda c_i}{C}\Bigg).$$

Substituting this into the original expression produces

$$\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} \geq \left[\prod_i \exp\left(\frac{\lambda c_i}{C} \left(1 + \frac{C}{\lambda c_i}\phi_i(\theta) \right)^{-p} - \frac{\lambda c_i}{C} + p\phi_i(\theta) \right) \right. \\ \left. + \prod_i \exp\left(\frac{\lambda c_i}{C} \left(1 + \frac{C}{\lambda c_i}\phi_i(\theta') \right)^{-p} - \frac{\lambda c_i}{C} + p\phi_i(\theta') \right) \right]^{-\frac{1}{p}}.$$

Considering the term inside exp. Define a function $f(y) = \frac{\lambda c_i}{C}\left(1 + \frac{C}{\lambda c_i}y\right)^{-p} - \frac{\lambda c_i}{C} + py$ for $y \geq 0$. It is clear that $f(0) = 0$. The first derivative is

$$f'(y) = p + (-p)\left(1 + \frac{C}{\lambda c_i}y\right)^{-p-1}$$

which is also 0 at $y = 0$. The second and third derivatives are

$$f''(y) = (-p)(-p-1)\frac{C}{\lambda c_i}\left(1 + \frac{C}{\lambda c_i}y\right)^{-p-2}, \tag{D.4}$$

$$f'''(y) = (-p)(-p-1)(-p-2)\left(\frac{C}{\lambda c_i}\right)^2\left(1 + \frac{C}{\lambda c_i}y\right)^{-p-3}. \tag{D.5}$$

By Taylor series, we have

$$f(y) = f(0) + f'(0)y + \frac{f''(0)}{2!}y^2 + \frac{f'''(v)}{3!}y^3$$

where v is between 0 and y. By (D.5), we know that $f'''(v) \leq 0$, therefore since $y \geq 0$, we have

$$\begin{aligned} f(y) &\leq f(0) + f'(0)y + \frac{f''(0)}{2!}y^2 \\ &= \frac{f''(0)}{2!}y^2. \end{aligned}$$

Substituting $y = \phi_i(\theta)$ produces

$$\begin{aligned} f(\phi_i(\theta)) &\leq (-p)(-p-1)\frac{C}{\lambda c_i}\phi_i^2(\theta) \\ &\leq (-p)(-p-1)\frac{C}{\lambda c_i}c_i^2 M^2(\theta,\theta'). \end{aligned}$$

Similarly, we can get

$$f(\phi_i(\theta')) \leq p(p+1)\frac{C}{\lambda c_i}c_i^2 M^2(\theta,\theta').$$

Substituting these to the spectral ratio, we get

$$\begin{aligned}
\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} &\geq \left[2\prod_i \exp\left(p(p+1)\frac{C}{\lambda c_i}c_i^2 M^2(\theta,\theta')\right)\right]^{-\frac{1}{p}} \\
&= \left[2\exp\left(\sum_i p(p+1)\frac{C}{\lambda}c_i M^2(\theta,\theta')\right)\right]^{-\frac{1}{p}} \\
&= \left[2\exp\left(p(p+1)\frac{C^2}{\lambda}M^2(\theta,\theta')\right)\right]^{-\frac{1}{p}} \\
&= 2^{-\frac{1}{p}}\exp\left(-(p+1)\frac{C^2}{\lambda}M^2(\theta,\theta')\right).
\end{aligned}$$

Now, we maximize the R.H.S. with respect to p. Let $E = \frac{C^2}{\lambda}M^2(\theta,\theta')$, then it becomes

$$2^{-\frac{1}{p}}\exp\left(-(p+1)E\right) = \exp\left(-E - pE - \frac{1}{p}\log 2\right).$$

The maximum is attained at $p = \sqrt{\frac{\log 2}{E}}$ and the value is

$$\exp\left(-E - 2\sqrt{E\log 2}\right).$$

It follows that

$$\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} \geq \exp\left(-\frac{C^2}{\lambda}M^2(\theta,\theta') - 2\sqrt{\frac{C^2}{\lambda}M^2(\theta,\theta')\log 2}\right).$$

We set $\lambda = \chi C^2 M^2(\theta,\theta')$, it becomes

$$\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} \geq \exp\left(-\frac{1}{\chi} - 2\sqrt{\frac{\log 2}{\chi}}\right).$$

We complete the theorem by a Dirichlet form argument. We can write the Dirichlet form $\mathcal{E}(f)$ of a Markov chain with transition operator G as [217]:

$$\mathcal{E}(f) = \frac{1}{2}\int\int\left[(f(\theta) - f(\theta'))^2\right]G(\theta,\theta')\pi(\theta)d\theta d\theta'.$$

If we let $L_0^2(\pi)$ to be the Hilbert space of functions f such that f has mean zero and is square integrable with respect to probability measure π. It follows that the spectral gap γ of a Markov chain is [15]

$$\gamma = \inf_{f \in L_0^2(\pi): Var_\pi[f]=1} \mathcal{E}(f).$$

From this, it is easy to get that

$$\begin{aligned}
\bar{\gamma} &= \inf_{f \in L_0^2(\pi): Var_\pi[f]=1} \left[\frac{1}{2} \int \int \left[(f(\theta) - f(\theta'))^2 \right] T(\theta, \theta') \pi(\theta) d\theta d\theta' \right] \\
&\geq \exp\left(-\frac{1}{\chi} - 2\sqrt{\frac{\log 2}{\chi}} \right) \cdot \inf_{f \in L_0^2(\pi): Var_\pi[f]=1} \left[\frac{1}{2} \int \int \left[(f(\theta) - f(\theta'))^2 \right] G(\theta, \theta') \pi(\theta) d\theta d\theta' \right] \\
&= \exp\left(-\frac{1}{\chi} - 2\sqrt{\frac{\log 2}{\chi}} \right) \cdot \gamma.
\end{aligned}$$

□

D.6 Derivation of Equation (5.2)

Based on the bound in Theorem 3, to make sure that the spectral ratio $\bar{\gamma}/\gamma \geq \kappa$, we can set χ such that

$$\exp\left(-\frac{1}{\chi} - 2\sqrt{\frac{\log 2}{\chi}} \right) = \kappa.$$

Solving the above equation gives us

$$\chi = \frac{(2\log 2 - \log \kappa + 2\sqrt{\log 2(\log 2 - \log \kappa)})}{\log^2 \kappa} \leq \frac{4}{(1-\kappa)\log(1/\kappa)}.$$

Since the spectral gap ratio is monotonically increasing w.r.t. χ, we can instead set χ to the upper bound

$$\chi = \frac{4}{(1-\kappa)\log(1/\kappa)}$$

D.7 Theoretically Optimal Value of χ

The overall wall-clock time L for a chain to converge can be represented as the number of steps times the wall-clock time l of each step. We then minimize an upper bound of this overall wall-clock time to get the optimal value of χ.

Consider a lazy Markov chain on a finite state Θ. The *relaxation time* t_{rel} of a Markov chain is defined to be the inverse of the spectral gap γ: $t_{\text{rel}} = 1/\gamma$. The *mixing time* t_{mix}, i.e. the number of steps required for a chain to converge to within TV distance δ to the target distribution π, is bounded by Levin and Peres [365]

$$t_{\text{mix}} \leq t_{\text{rel}} \log\left(\frac{1}{\delta \cdot \min_{\theta\in\Theta} \pi(\theta)}\right).$$

It follows that the overall wall-clock time L is upper bounded by

$$L = l \cdot t_{\text{mix}} \leq l \cdot t_{\text{rel}} \log\left(\frac{1}{\delta \cdot \min_{\theta\in\Theta} \pi(\theta)}\right).$$

We assume that the expected wall clock time to run a step is proportional to the batch size plus some constant, which measures the cost of computing the proposal. Specifically, We use η and ξ to denote the time to get a proposal θ' and compute a U_i in a step. Then we can write the time of a step l as

$$l = B\xi + \eta.$$

In order to minimize L, we can instead minimize its upper bound, which is equivalent to minimize

$$l \cdot t_{\text{rel}} = (B\xi + \eta) \cdot \frac{1}{\gamma}. \tag{D.6}$$

Recall that for TunaMH, the average batch size over all steps is

$$\mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[\chi C^2 M^2(\theta,\theta') + CM(\theta,\theta')],$$

and the spectral gap $\bar{\gamma}$ is lower bounded by the spectral gap of standard MH γ such that

$$\bar{\gamma} \geq \exp\left(-\frac{1}{\chi} - 2\sqrt{\frac{\log 2}{\chi}}\right)\cdot\gamma.$$

Substituting the expression of batch size and spectral gap to (D.6) gives

$$l\cdot t_{\text{rel}} \leq \left(\mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[\chi C^2 M^2(\theta,\theta') + CM(\theta,\theta')]\xi + \eta\right)\cdot\exp\left(\frac{1}{\chi} + 2\sqrt{\frac{\log 2}{\chi}}\right)\cdot\frac{1}{\gamma}.$$

To minimize the RHS of the above equation over χ, we let the derivative w.r.t. χ to be zero and get,

$$\begin{aligned}
&\xi C^2\mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M^2(\theta,\theta')]\chi^{-1} + (\xi C\mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M(\theta,\theta')] + \eta)\chi^{-2}\\
&\quad + \sqrt{\log 2}\xi C^2\mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M^2(\theta,\theta')]\chi^{-\frac{1}{2}}\\
&\quad + \sqrt{\log 2}(\xi C\mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M(\theta,\theta')] + \eta)\chi^{-\frac{3}{2}}\\
&\quad = \xi C^2\mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M^2(\theta,\theta')].
\end{aligned}$$

When χ is small, the LHS is approximately $(\xi C\mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M(\theta,\theta')] + \eta)\chi^{-2}$ which gives us

$$\chi = \sqrt{\frac{\xi C\mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M(\theta,\theta')] + \eta}{\xi C^2\mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M^2(\theta,\theta')]}}.$$

When it is quick to get a proposal ($\eta \approx 0$) and the variance of M is small, we can further simplify it to

$$\chi = \frac{1}{\sqrt{C\mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M(\theta,\theta')]}}.$$

In practice, we can get the above theoretically optimal value of χ by empirically estimating the mean and variance of $M(\theta, \theta')$. Note that even if these empirical estimates are accurate, there may exist better χ, since the upper bounds (the mixing time bound and the spectral gap bound) we use to get the optimal value may be loose. We give a simpler heuristic to tune χ in practice in Section 5.5.

D.8 Proof of Theorem 4

First, we will show the following lemma, which gives half of what we want to have in the theorem.

Lemma 1. *Considering the same setting as the theorem, the average batch size B of any exact, stateless minibatch MH algorithm at any iteration follows*

$$\mathbf{E}[B] \geq 2^{-18} \cdot \kappa C^2 M^2(\theta, \theta') - 2^{-4} \cdot \kappa.$$

Proof. We prove the lemma by construction. First, observe that since the state space Θ has at least two states, we can restrict our attention to just two of those states, by choosing a π that has zero mass on any other state in the space and a q that never proposes transitioning out to any of those other states (at which π has zero mass). Such a proposal will still be ergodic, so it still satisfies our general assumption that we consider only ergodic chains in this paper. Without loss of generality, suppose that those two states are $\{-\frac{M}{2}, \frac{M}{2}\}$ (this is without loss of generality because we can always just rename the states), and let C denote the constant in the theorem statement and define (with a bit of abuse of notation) the constant $M := M(-\frac{M}{2}, \frac{M}{2})$. By doing this, we can (again without loss of generality) restrict our attention to the case where $\Theta = \{-\frac{M}{2}, \frac{M}{2}\}$.

Next, we construct our counterexample. Let the dataset be $\{x_i\}_{i=1}^N$ where $x_i \in \{-1, 1\}$. We let the domain for parameter θ to be $\{-\frac{M}{2}, \frac{M}{2}\}$, and the target distribution to be

$$\pi(\theta) = \frac{1}{Z} \exp\left(-\sum_{i=1}^{N} U_i(\theta)\right) = \frac{1}{Z} \exp\left(-\frac{C\theta}{N} \sum_{i=1}^{N} x_i\right)$$

where $U_i(\theta) = \frac{C}{N} \cdot \theta x_i$. Note that by letting N become large, any minibatch MH algorithm that queries the energy difference oracle some number of times will observe a distribution of energy differences that is arbitrarily close to a sequence of independent identically distributed random variables supported on $\{\pm\frac{CM}{N}\}$.

We define $c_i = \frac{C}{N}$, and the proposal distribution to be

$$p(\theta, \theta) = \frac{1}{2}, \qquad p(\theta, -\theta) = \frac{1}{2} \qquad \text{for } \theta \in \left\{-\frac{M}{2}, \frac{M}{2}\right\}.$$

Now, let $0 < q < 1$ be some constant, and consider two cases: (1) $\frac{1}{N}\sum_i x_i = q$ and (2) $\frac{1}{N}\sum_i x_i = -q < 0$. Suppose that in both cases the x_i are shuffled at random. These two cases will have different stationary distributions,

$$\pi_1(\theta) = \frac{1}{Z} \exp\left(-Cq\theta\right) \qquad \text{and} \qquad \pi_2(\theta) = \frac{1}{Z} \exp\left(Cq\theta\right),$$

and an exact algorithm must be able to distinguish between them. Therefore by using these cases, we can get a bound on the required batch size needed for the exact MH algorithm to distinguish between them. First, we observe that the two cases are symmetric, such that if T_1 is the transition matrix of the chain in case (1) and T_2 is the transition matrix of the chain in case (2), then $T_1(\theta, \theta') = T_2(\theta', \theta)$. Let $0 < \psi < \frac{1}{2}$ denote the probability that T_1 transitions from $\frac{M}{2}$ to $-\frac{M}{2}$. Then because the MH method is exact and the chain is reversible, the probability of the reverse transition is $\psi \exp(-CMq)$. So, explicitly, the transition operators will look like

$$T_1 = \begin{bmatrix} 1-\psi & \psi e^{-CMq} \\ \psi & 1-\psi e^{-CMq} \end{bmatrix} \qquad \text{and} \qquad T_2 = \begin{bmatrix} 1-\psi e^{-CMq} & \psi \\ \psi e^{-CMq} & 1-\psi \end{bmatrix}.$$

The eigenvectors and eigenvalues of this are

$$T_1\pi_1 = \pi_1 \qquad \text{and} \qquad T_1\begin{bmatrix}-1\\ \\ 1\end{bmatrix} = (1-\psi-\psi\exp(-CMq))\begin{bmatrix}-1\\ \\ 1\end{bmatrix}.$$

Suppose that we initialize both chains uniformly on $\{-\frac{M}{2}, \frac{M}{2}\}$. Observe that

$$\begin{bmatrix}1/2\\ \\ 1/2\end{bmatrix} = \begin{bmatrix}\frac{\exp(-CMq)}{1+\exp(-CMq)}\\ \\ \frac{1}{1+\exp(-CMq)}\end{bmatrix} + \frac{1-\exp(-CMq)}{2(1+\exp(-CMq))}\cdot\begin{bmatrix}1\\ \\ -1\end{bmatrix},$$

the first vector being π_1 and the second being a multiple of the other eigenvector.

Equivalently,

$$\begin{bmatrix}1/2\\ \\ 1/2\end{bmatrix} = \pi_1 + \frac{1}{2}\tanh\left(\frac{CMq}{2}\right)\cdot\begin{bmatrix}1\\ \\ -1\end{bmatrix},$$

and so for any t, after t steps of the Markov chain, the distribution will be

$$T_1^t\begin{bmatrix}1/2\\ \\ 1/2\end{bmatrix} = \pi_1 + \frac{1}{2}\tanh\left(\frac{CMq}{2}\right)\cdot(1-\psi-\psi\exp(-CMq))^t\cdot\begin{bmatrix}1\\ \\ -1\end{bmatrix}.$$

Similarly,

$$T_2^t\begin{bmatrix}1/2\\ \\ 1/2\end{bmatrix} = \pi_2 + \frac{1}{2}\tanh\left(\frac{CMq}{2}\right)\cdot(1-\psi-\psi\exp(-CMq))^t\cdot\begin{bmatrix}-1\\ \\ 1\end{bmatrix}.$$

So, the total variation distance between the state of the chains at time t will be bounded by

$$\mathrm{TV}\left(T_1^t\begin{bmatrix}1/2\\ \\ 1/2\end{bmatrix}, T_2^t\begin{bmatrix}1/2\\ \\ 1/2\end{bmatrix}\right) \geq \mathrm{TV}(\pi_1,\pi_2) - \tanh\left(\frac{CMq}{2}\right)\cdot(1-\psi-\psi\exp(-CMq))^t.$$

Also observe that

$$\mathrm{TV}(\pi_1,\pi_2) = \frac{1}{2}\left\|\begin{bmatrix}\frac{\exp(-CMq)}{1+\exp(-CMq)}\\ \\ \frac{1}{1+\exp(-CMq)}\end{bmatrix} - \begin{bmatrix}\frac{1}{1+\exp(-CMq)}\\ \\ \frac{\exp(-CMq)}{1+\exp(-CMq)}\end{bmatrix}\right\|_1 = \frac{1-\exp(-CMq)}{1+\exp(-CMq)} = \tanh\left(\frac{CMq}{2}\right),$$

so

$$\mathrm{TV}\left(T_1^t\begin{bmatrix}1/2\\1/2\end{bmatrix}, T_2^t\begin{bmatrix}1/2\\1/2\end{bmatrix}\right) \geq \tanh\left(\frac{CMq}{2}\right)\cdot\left(1-(1-\psi-\psi\exp(-CMq))^t\right).$$

Also, since we know that our algorithm is guaranteed to have spectral gap ratio at least κ with the original chain, it follows that $\psi \geq \kappa/2$, and so

$$\mathrm{TV}\left(T_1^t\begin{bmatrix}1/2\\1/2\end{bmatrix}, T_2^t\begin{bmatrix}1/2\\1/2\end{bmatrix}\right) \geq \tanh\left(\frac{CMq}{2}\right)\cdot\left(1-\left(1-\frac{\kappa}{2}-\frac{\kappa}{2}\exp(-CMq)\right)^t\right).$$

Now, denote the exact minibatch algorithm to be $\mathcal{A}$. As it runs, the algorithm $\mathcal{A}$ will request data examples by querying the energy difference oracle. Under case (1), we let y_i denote the ith sample that $\mathcal{A}$ *would have observed* if it requested i or more samples, and similarly we let z_i denote the analogous sample in case (2). Fix some constant $t \in \mathbf{N}$ (which we will set later). We let K_1 denote the total number of samples observed by $\mathcal{A}$ across the first t iterations in case (1), and set

$$\mu = \{y_1, y_2, \ldots, y_{K_1}\}.$$

Similarly, we let K_2 denote the number of samples observed by $\mathcal{A}$ across the first t iterations in case (2), and set

$$\nu = \{z_1, z_2, \ldots, z_{K_2}\}.$$

Now, we fix some constant K (to be set later), and consider the following coupling between the behavior of $\mathcal{A}$ across its first t iterations in case (1) and in case (2). First, let all internal randomness of $\mathcal{A}$ and the proposal process under case (1) and (2) be the same, which means that for a given observation of data examples, the algorithm $\mathcal{A}$ will make the same decision, such as whether to require more data examples or not and whether to accept or not. Second, choose a coupling that minimizes the probability that

$$(y_1, y_2, \ldots, y_{K1}) \neq (z_1, z_2, \ldots, z_{K2}).$$

Such a coupling is guaranteed to exist by the Coupling Lemma, and the probability that these two are not equal will be equal to the total variation distance between their distributions. Third, assign all the other y_i and z_i, for $i > K$, independently according to their distribution.

We are interested in the quantity $p(\mu \neq \nu)$, which bounds the probability that the algorithm may make a different decision in cases (1) and (2). We can decompose this probability into two terms,

$$p(\mu \neq \nu) = p(\mu \neq \nu \text{ and } y_j = z_j \text{ for all } j \leq K) + p(\mu \neq \nu \text{ and } y_j \neq z_j \text{ for some } j \leq K).$$

If $\mu \neq \nu$ but $y_j = z_j$ for all $j \leq K$, the only way that this is possible is for $K_1 > K$ (and, symmetrically, also $K_2 > K$), since otherwise the algorithms would behave identically. So,

$$p(\mu \neq \nu) \leq p(K_1 > K) + p(y_j \neq z_j \text{ for some } j \leq K). \tag{D.7}$$

By Markov's inequality,

$$p(\mu \neq \nu) \leq \frac{\mathbf{E}[K_1]}{K} + p(y_j \neq z_j \text{ for some } j \leq K).$$

For the second term of (D.7), we can reduce the case to only considering K samples. Let S_y be the total number of samples y_i that are -1 and let S_z be the total number of samples z_i that are -1. Since $\mathcal{A}$ is effectively sampling a shuffled dataset at some arbitrary indices without replacement, both of these random variables S_y and S_z are—properly speaking—hypergeometric random variables. However, since our dataset size N is arbitrary here, we can by setting N very large work in the limit (as $N \to \infty$) in which these variables become binomial (since sampling with replacement and without replacement can be made to have arbitrarily close to the same distribution by making the dataset large). Observe that (in this limit) S_y follows a binomial distribution $B(K, \frac{1-q}{2})$ and S_z

follows a binomial distribution $B(K, \frac{1+q}{2})$. Clearly, if $S_y = S_z$, then we can arrange the coupling so that $(y_1, \ldots, y_K) = (z_1, \ldots, z_K)$. So, by the Coupling Lemma,

$$p(y_j \neq z_j \text{ for some } j \leq K) = p(S_y \neq S_z) = \text{TV}(S_y, S_z).$$

From the analysis in Adell and Jodrá [8], we can bound the total variance distance between these two binomial variables with

$$\text{TV}(S_y, S_z) \leq \sqrt{e} \cdot \frac{\tau}{(1-\tau)^2}$$

where $\tau = \sqrt{\frac{K+2}{2}} \cdot q < 1$. Substituting these bounds, we get

$$p(\mu \neq \nu) \leq \frac{\mathbf{E}[K_1]}{K} + \sqrt{e} \cdot \frac{\tau}{(1-\tau)^2}.$$

But the probability that $\mu \neq \nu$ must be an upper bound on the probability that the distributions of the chains in case (1) and (2) after t steps are not equal, since if $\mu = \nu$ in the coupling then the two chains are in the same state. So, using our bound from earlier, we get

$$\tanh\left(\frac{CMq}{2}\right) \cdot \left(1 - \left(1 - \frac{1}{2}\kappa - \frac{1}{2}\kappa \exp(-CMq)\right)^t\right) \leq \frac{\mathbf{E}[K_1]}{K} + \sqrt{e} \cdot \frac{\tau}{(1-\tau)^2}.$$

Now isolating $\mathbf{E}[K_1]$ gives

$$K \cdot \tanh\left(\frac{CMq}{2}\right) \cdot \left(1 - \left(1 - \frac{1}{2}\kappa - \frac{1}{2}\kappa \exp(-CMq)\right)^t\right) - K \cdot \sqrt{e} \cdot \frac{\tau}{(1-\tau)^2} \leq \mathbf{E}[K_1].$$

Also, observe that

$$\left(1 - \frac{1}{2}\kappa - \frac{1}{2}\kappa \exp(-CMq)\right)^t \leq \left(1 - \frac{1}{2}\kappa\right)^t \leq \exp\left(-\frac{\kappa t}{2}\right),$$

so

$$K \cdot \tanh\left(\frac{CMq}{2}\right) \cdot \left(1 - \exp\left(-\frac{\kappa t}{2}\right)\right) - K \cdot \sqrt{e} \cdot \frac{\tau}{(1-\tau)^2} \leq \mathbf{E}[K_1].$$

This gives us the lower bound on $\mathbf{E}[K_1]$ that we are interested in. Now, it remains to assign q, K, and t. We start by assigning t such that

$$t = \left\lceil 2\kappa^{-1} \log(2) \right\rceil,$$

in which case

$$\exp\left(-\frac{\kappa t}{2}\right) \leq \frac{1}{2}$$

and so

$$K \cdot \frac{1}{2} \cdot \tanh\left(\frac{CMq}{2}\right) - K \cdot \sqrt{e} \cdot \frac{\tau}{(1-\tau)^2} \leq \mathbf{E}[K_1].$$

Now, we add some simplifying assumptions, which we will validate are true later. We assume that

$$\tau = \sqrt{\frac{K+2}{2}} \cdot q \leq \frac{1}{2};$$

in this case

$$\sqrt{e} \cdot \frac{\tau}{(1-\tau)^2} \cdot K \leq 4\sqrt{e} \cdot \tau \leq 5\sqrt{K+2} \cdot q.$$

We set q such that

$$CMq = 1,$$

and we assume that CM is large enough that this assignment of q is within range (i.e. $0 < q < 1$). This gives us

$$K \cdot \frac{1}{2} \cdot \tanh\left(\frac{1}{2}\right) - 5K\sqrt{K+2} \cdot \frac{1}{CM} \leq \mathbf{E}[K_1].$$

Since $\tanh(1/2) > 5/16$, we can simplify this to

$$K \cdot \frac{5}{32} - 5K\sqrt{K+2} \cdot \frac{1}{CM} \leq \mathbf{E}[K_1].$$

All that remains is to assign K. We assign K such that

$$\sqrt{K+2} \cdot \frac{1}{CM} = \frac{1}{64}.$$

In this case, we get

$$K = \frac{C^2M^2}{4096} - 2,$$

and our bound reduces to

$$\left(\frac{C^2M^2}{4096} - 2\right) \cdot \frac{5}{64} \leq \mathbf{E}[K_1].$$

We can simplify this further to

$$2^{-16} \cdot C^2M^2 - \frac{5}{32} \leq \mathbf{E}[K_1].$$

Now, this is a bound on the expected number of samples taken across t iterations. This means that the number of samples taken in any given iteration will be bounded by

$$\frac{\mathbf{E}[K_1]}{t} \geq \frac{2^{-16} \cdot C^2M^2 - \frac{5}{32}}{2\kappa^{-1}\log(2) + 1} = \frac{2^{-16} \cdot \kappa C^2M^2 - \frac{5\kappa}{32}}{2\log(2) + \kappa}.$$

A few more loose bounds, leveraging $\kappa < 1$, gives us

$$\frac{\mathbf{E}[K_1]}{t} \geq 2^{-18} \cdot \kappa C^2M^2 - \frac{\kappa}{16}.$$

This proves the lemma. □

Next, we will show the following lemma, which characterizes what happens when CM is small.

Lemma 2. *Considering minibatch MH algorithms in the same setting as the theorem, the expected batch size at any iteration must be lower bounded by*

$$\mathbf{E}[B] \geq \frac{\kappa}{2} \min\left(CM(\theta, \theta'), 1\right).$$

Proof. Here, we will prove a lower bound that characterizes the limits of exact stateless minibatch MH algorithms when they use very few examples. Again,

without loss of generality we consider a reduction to the two-state case as we did in the proof of the previous lemma. Suppose that a exact stateless mini-batch MH algorithm with the same forward and backward proposal probabilities (given some $c_1, \ldots, c_N$, C, and M) requests any energy function examples at all only with probability p. Consider two cases, which have the same $c_1, \ldots, c_N$, C and M. In the first case,

$$\sum_{i=1}^{n} (U_i(\theta) - U_i(\theta')) = CM(\theta, \theta'),$$

while in the second case,

$$\sum_{i=1}^{n} (U_i(\theta) - U_i(\theta')) = -CM(\theta, \theta').$$

These are clearly possible by setting U_i to the limits of what is covered by the bounds. In the first case, the baseline MH method would accept with probability 1. In the second case, it will accept with probability $\exp(-CM(\theta, \theta'))$. Since the stateless MH algorithm is reversible, it must accept in the first case with some probability a and in the second case with probability $a \cdot \exp(-CM(\theta, \theta'))$. But, the algorithm can only distinguish the two cases if it requests samples, which only happens with probability at most p. So,

$$a - a \cdot \exp(-CM(\theta, \theta')) \leq p.$$

Since we know that it must be the case that $a \geq \kappa$ (from a straightforward analysis of a two-state case), it follows that

$$\frac{p}{\kappa} \geq \frac{p}{a} \geq 1 - \exp(-CM(\theta, \theta')) \geq \frac{1}{2} \min\left(CM(\theta, \theta'), 1\right).$$

Since p is an obvious lower bound on the expected value of the batch size, it follows that

$$\mathbf{E}[B] \geq \frac{\kappa}{2} \min\left(CM(\theta, \theta'), 1\right).$$

□

To prove Theorem 4 we now combine the results of these two lemmas. We have

$$\mathbf{E}[B] \geq 2^{-18} \cdot \kappa C^2 M^2(\theta, \theta') - 2^{-4} \cdot \kappa.$$

and

$$\mathbf{E}[B] \geq \frac{\kappa}{2} \min\left(CM(\theta, \theta'), 1\right).$$

Since these are both lower bounds, we can combine them to get

$$\begin{aligned}\mathbf{E}[B] &\geq \max\left(2^{-18} \cdot \kappa C^2 M^2(\theta, \theta') - 2^{-4} \cdot \kappa, \frac{\kappa}{2} \min\left(CM(\theta, \theta'), 1\right)\right) \\ &= \kappa \cdot \max\left(2^{-18} \cdot C^2 M^2(\theta, \theta') - 2^{-4}, \frac{1}{2} \min\left(CM(\theta, \theta'), 1\right)\right).\end{aligned}$$

It is obvious from a simple big-O analysis here that there exists a global constant $\zeta > 0$ such that

$$\mathbf{E}[B] \geq \zeta \cdot \kappa\left(C^2 M^2(\theta, \theta') + CM(\theta, \theta')\right).$$

This proves the theorem.

D.9 Proof of Corollary 1

Proof. Recall that the lower bound on the batch size in each iteration is

$$\mathbf{E}[B] \geq \zeta \cdot \kappa\left(C^2 M^2(\theta, \theta') + CM(\theta, \theta')\right).$$

Since $C = \Theta(N)$ and $M(\theta, \theta') = \Theta(N^{-(h+1)/2})$, the expectation of the batch size follows

$$\mathbf{E}[B] = \Theta(C^2 M^2(\theta, \theta') + CM(\theta, \theta')) = \Theta(CM(\theta, \theta')) = \Theta(N^{1-h}/2).$$

When $h = 1$, $\mathbf{E}[B] = \Theta(1)$ and when $h = 2$, $\mathbf{E}[B] = \Theta(1/\sqrt{N})$. □

D.10 Experimental Details and Additional Results

D.10.1 Experiment in Section 5.2.1

To verify Theorem 2, we empirically construct a distribution in the form of Section D.1, on which AustereMH and MHminibatch are biased. Note that the proof in Section D.1 shows there must exist such a distribution for any inexact minibatch method but does not tell us how to find one for a specific method. Therefore, in order to find such a distribution, we construct an example and empirically test whether AustereMH and MHminibatch are biased on it.

We let data x_i take one of two values $\{-1, 5\}$. Consider a dataset of size 6000. We let 5000 data take value -1 and the remaining 1000 data take value 5. Define the target distribution $\pi(\theta)$ to be

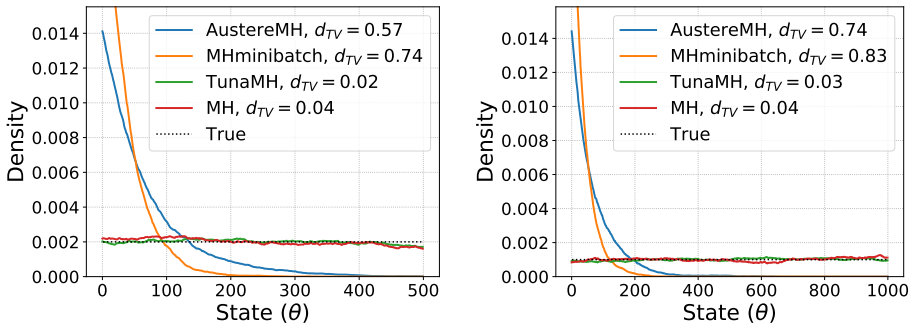
$$\pi(\theta) \propto \exp\left(-\frac{1}{N}\sum_{i=1}^{N} \theta \cdot x_i\right)$$

where the domain of θ is $\{0, 1, \ldots, K-1\}$. Therefore the number of state is K. Since $\sum_i x_i = 0$, it is clear to see that the stationary distribution of θ is a uniform distribution. We define the proposal distribution to be the following

$$p(\theta, \theta) = \frac{1}{2}, \quad \text{for all } \theta; \quad p(\theta, \theta-1) = \frac{1}{4}, \quad p(\theta, \theta+1) = \frac{1}{4} \quad \text{for } \theta \in \{1, \ldots, K-2\};$$

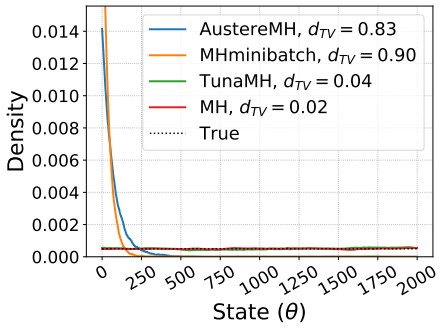
and $p(0, 1) = p(K-1, K-2) = \frac{1}{2}$.

We set the hyperparameter error ϵ in AustereMH to be 0.01 and δ in MHminibatch to be 5, following the setting in their original papers [327, 528]. We set batch size m in both methods to be 30. We find that AustereMH and MHminibatch are both inexact on this example and the error increases as we increase K. Thus we empirically verify the statement in Theorem 2.

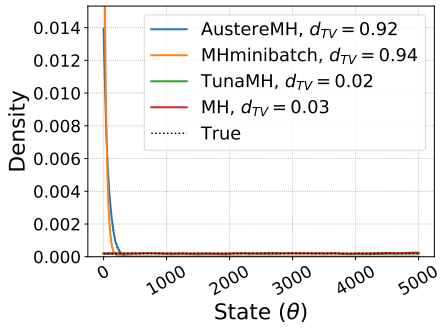


(a) $K = 500$

(b) $K = 1000$



(e) $K = 2000$



(e) $K = 5000$

Figure D.1: Density estimate comparison on $K = 500, 1000, 2000, 5000$.

Besides the density estimate comparison on $K = 200$ shown in Figure 5.1b, we additionally report the estimate results on other values of K in Figure D.1. We see that the results are similar, all showing that TunaMH and standard MH can give accurate estimate whereas inexact methods are seriously wrong.

Robust Linear Regression We further tested AustereMH on robust linear regression in Section 5.5.1 with $N = 5000$. We computed the MSE between estimated and true parameters. MH, TunaMH and AustereMH obtained MSE 0.149, 0.15 and 1.19 respectively, indicating inexact method error can be large on typical problems.

D.10.2 Robust Linear Regression

We follow the experimental setup of robust linear regression (RLR) in Cornish et al. [155]. Specifically, we have data $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. The likelihood is modeled by a student's t-distribution with degrees of freedom v:

$$p(y_i|\theta, x_i) = \text{Student}(y_i - \theta^\intercal x_i|v).$$

It follows that

$$U_i(\theta) = \frac{v+1}{2}\log\left(1 + \frac{(y_i - \theta^\intercal x_i)^2}{v}\right),$$

and the first derivative

$$\partial_j U_i(\theta) = -(v+1)\frac{x_{ij}(y_i - \theta^\intercal x_i)}{v + (y_i - \theta^\intercal x_i)^2}.$$

Since the function U_i is Lipschitz continuous, we can easily get the bound used in TunaMH, TFMH and SMH. We set $M(\theta, \theta') = \|\theta - \theta'\|_2$ and then it follows

$$c_i = \sup_{\theta\in\mathbb{R}} \|\nabla U_i(\theta)\|_2 = \frac{v+1}{2\sqrt{v}}\|x_i\|_2.$$

The data x_i and y_i is generated as follows

$$y_i = \sum_j x_{ij} + \epsilon_i$$

where $\epsilon_i \sim \mathcal{N}(0, 1)$.

In Section 5.5.1, we set $v = 4$, $d = 100$ and use a flat prior $p(\theta) = 1$. Note that our problem dimension d is much larger than that in the SMH paper [155] ($d = 10$). This makes the control variates in SMH problematic since the bounds they require appear to scale badly in high dimensions.

To reach the target acceptance rate, we set the stepsize in each method as in Table D.1 and D.2. For TunaMH and TunaMH-MAP, we set $\chi = 1e-5$ for

Table D.1: Stepsize of methods without the MAP.

	MH	TFMH	FlyMC	TunaMH
RLR $N = 5000$	4e-3	1e-4	2.7e-3	8e-4, $\chi = 1e-5$
RLR $N = 20000$	2e-3	3e-5	1.5e-3	3e-4, $\chi = 1e-5$
RLR $N = 50000$	1.3e-3	1.2e-5	9e-4	2e-4, $\chi = 1e-4$
RLR $N = 100000$	9e-4	6e-6	7e-4	1.7e-4, $\chi = 1e-4$
TGM	3e-1	2.2e-2	1e-2	1e-1
LR	5e-3	1e-4	2e-3	1e-3

$N = 5000, 20000$ and $\chi = 1e-4$ for $N = 50000, 100000$. For FlyMC and FlyMC-MAP, we set the probability for a data going from dark to bright $q_{d\to b}$ to be 0.01. Without the MAP, we collect 80000 samples after 200000 step burnin. With the MAP, we collect 80000 samples without burnin.

Table D.2: Stepsize of methods with the MAP.

	MH-MAP	SMH-1	SMH-2	FlyMC-MAP	TunaMH-MAP
RLR $N = 5000$	4e-3	4e-3	4e-3	6e-3	8e-4, $\chi = 1e-5$
RLR $N = 20000$	2e-3	2e-3	2e-3	3.5e-3	3e-4, $\chi = 1e-5$
RLR $N = 50000$	1.2e-3	1.2e-3	1.2e-3	2.5e-3	1.2e-4, $\chi = 1e-4$
RLR $N = 100000$	9e-4	5.9e-4	8e-4	1.7e-3	7e-5. $\chi = 1e-4$
TGM	-	1e-1	-	1e-2	-

Additional Experimental Results with $d = 10$

We ran RLR experiment with $d = 10$ and $N = 10^5$ to compare the performance in low dimensions. The ESS/S for TFMH, FlyMC, TunaMH are 0.02, 0.75, & 1.7, respectively; SMH-1, SMH-2, FlyMC-MAP and TunaMH-MAP are 174.7, 5969.5, 730.8, & 730.1 respectively. This suggests TunaMH is significantly better without MAP/control variates. With MAP/control variates, TunaMH is better

than SMH-1, similar to FlyMC and worse than SMH-2.

D.10.3 Truncated Gaussian Mixture

The data in this truncated Gaussian mixture (TGM) task is generated as follows

$$x_i \sim \frac{1}{2}\mathcal{N}(\theta_1, \sigma_x^2) + \frac{1}{2}\mathcal{N}(\theta_1 + \theta_2, \sigma_x^2)$$

where $\theta_1 = 0, \theta_2 = 1$ and $\sigma^2 = 2$. The posterior θ has two modes at $(\theta_1, \theta_2) = (0, 1)$ and $(\theta_1, \theta_2) = (1, -1)$. In order to get the bounds required by all methods, we truncate the Gaussian by setting $\theta_1, \theta_2 \in [-3, 3]$.

For simplicity we assume a flat prior $p(\theta) = 1$. Then the energy is given by

$$U_i(\theta) = -\log p(x_i|\theta) = \log\Big(2\sqrt{2\pi}\sigma_x\Big) - \log\bigg[\exp\Big(-\frac{(x_i-\theta_1)^2}{2\sigma_x^2}\Big) + \exp\Big(-\frac{(x_i-\theta_1-\theta_2)^2}{2\sigma_x^2}\Big)\bigg].$$

Denote $E_1 = \exp\Big(-\frac{(x_i-\theta_1)^2}{2\sigma_x^2}\Big)$ and $E_2 = \exp\Big(-\frac{(x_i-\theta_1-\theta_2)^2}{2\sigma_x^2}\Big)$. To get the upper bound in TunaMH, TFMH and SMH, we compute the gradient

$$\begin{aligned}
\frac{\partial U_i(\theta)}{\partial \theta_1} &= -\frac{1}{E_1 + E_2}\Big(E_1 \cdot \frac{x_i - \theta_1}{\sigma_x^2} + E_2 \cdot \frac{x_i - \theta_1 - \theta_2}{\sigma_x^2}\Big), \\
\frac{\partial U_i(\theta)}{\partial \theta_2} &= -\frac{1}{E_1 + E_2}\Big(E_2 \cdot \frac{x_i - \theta_1 - \theta_2}{\sigma_x^2}\Big).
\end{aligned}$$

Since $\theta_i \in [-3, 3]$, it follows that

$$\begin{aligned}
\Big|\frac{\partial U_i(\theta)}{\partial \theta_1}\Big| &\le \frac{|x_i| + 3}{\sigma_x^2} + \frac{|x_i| + 3 + 3}{\sigma_x^2} \le \frac{2\,|x_i| + 9}{\sigma_x^2}, \\
\Big|\frac{\partial U_i(\theta)}{\partial \theta_2}\Big| &\le \frac{|x_i| + 3 + 3}{\sigma_x^2} \le \frac{|x_i| + 6}{\sigma_x^2}.
\end{aligned}$$

Therefore we can set $M(\theta, \theta') = \|\theta - \theta'\|_2$ and

$$c_i = \sqrt{\Big(\frac{2\,|x_i| + 9}{\sigma_x^2}\Big)^2 + \Big(\frac{|x_i| + 6}{\sigma_x^2}\Big)^2}.$$

To use the control variate in SMH, we need to compute the second derivatives

$$\begin{aligned}
\frac{\partial^2 U_i(\theta)}{\partial^2 \theta_1} &= \frac{1}{(E_1+E_2)^2} \cdot \left(E_1 \cdot \frac{x_i-\theta_1}{\sigma_x^2} + E_2 \cdot \frac{x_i-\theta_1-\theta_2}{\sigma_x^2}\right)^2 \\
&\quad - \left[E_1 \cdot \left(\left(\frac{x_i-\theta_1}{\sigma_x^2}\right)^2 - \frac{1}{\sigma_x^2}\right) + E_2 \cdot \left(\left(\frac{x_i-\theta_1-\theta_2}{\sigma_x^2}\right)^2 - \frac{1}{\sigma_x^2}\right)\right] \cdot \frac{1}{E_1+E_2} \\
\frac{\partial^2 U_i(\theta)}{\partial \theta_1 \partial \theta_2} &= \frac{1}{(E_1+E_2)^2} \cdot \left(E_2 \cdot \left(\frac{x_i-\theta_1-\theta_2}{\sigma_x^2}\right)\right) \cdot \left(E_1 \cdot \frac{x_i-\theta_1}{\sigma_x^2} + E_2 \cdot \frac{x_i-\theta_1-\theta_2}{\sigma_x^2}\right) \\
&\quad - \left[E_2 \left(\left(\frac{x_i-\theta_1-\theta_2}{\sigma_x^2}\right)^2 - \frac{1}{\sigma_x^2}\right)\right] \cdot \frac{1}{E_1+E_2} \\
\frac{\partial^2 U_i(\theta)}{\partial^2 \theta_2} &= \frac{1}{(E_1+E_2)^2} \cdot \left(E_1 \cdot \frac{x_i-\theta_1}{\sigma_x^2} + E_2 \cdot \frac{x_i-\theta_1-\theta_2}{\sigma_x^2}\right)^2 \\
&\quad - \left[E_2 \cdot \left(\left(\frac{x_i-\theta_1-\theta_2}{\sigma_x^2}\right)^2 - \frac{1}{\sigma_x^2}\right)\right] \cdot \frac{1}{E_1+E_2}.
\end{aligned}$$

Given the parameter space, we have the upper bounds

$$\begin{aligned}
\left|\frac{\partial^2 U_i(\theta)}{\partial^2 \theta_1}\right| &\le \left(\frac{2|x_i|+9}{\sigma_x^2}\right)^2 + \left(\frac{|x_i|+3}{\sigma_x^2}\right)^2 + \left(\frac{|x_i|+6}{\sigma_x^2}\right)^2 + \frac{2}{\sigma_x^2} \\
\left|\frac{\partial^2 U_i(\theta)}{\partial \theta_1 \partial \theta_2}\right| &\le \frac{2|x_i|+9}{\sigma_x^2} \cdot \frac{|x_i|+6}{\sigma_x^2} + \left(\frac{|x_i|+6}{\sigma_x^2}\right)^2 + \frac{1}{\sigma_x^2} \\
\left|\frac{\partial^2 U_i(\theta)}{\partial^2 \theta_2}\right| &\le \left(\frac{2|x_i|+9}{\sigma_x^2}\right)^2 + \left(\frac{|x_i|+6}{\sigma_x^2}\right)^2 + \frac{1}{\sigma_x^2}.
\end{aligned}$$

It follows

$$\bar{U}_{2,i} = \left(\frac{2|x_i|+9}{\sigma_x^2}\right)^2 + \left(\frac{|x_i|+3}{\sigma_x^2}\right)^2 + \left(\frac{|x_i|+6}{\sigma_x^2}\right)^2 + \frac{2}{\sigma_x^2}.$$

which is required in SMH-1.

To get the lower bounds in FlyMC, we use the first-order Taylor expansion for $U_i(\theta)$. Higher order approximation is possible but would require heavier computation. By Taylor expansion,

$$U_i(\theta) = U_i(\theta^0) + \nabla U_i(\theta^0)^\intercal(\theta-\theta^0) + \frac{1}{2}(\theta-\theta^0)^\intercal \nabla^2 U_i(c)(\theta-\theta^0)$$

where c is between θ and θ^0.

Then we can define $\log B_i(\theta)$ in FlyMC as the follows

$$\begin{aligned}\log B_i(\theta) &= -U_i(\theta^0) - \nabla U_i(\theta^0)^\intercal(\theta - \theta^0) - \frac{1}{2} \cdot \max_c \left\|\nabla^2 U_i(c)\right\|_1 \cdot \left\|\theta - \theta^0\right\|_1^2 \\ &= -U_i(\theta^0) - \nabla U_i(\theta^0)^\intercal(\theta - \theta^0) - \frac{1}{2} \cdot \bar{U}_{2,i} \cdot \left\|\theta - \theta^0\right\|_1^2.\end{aligned}$$

The sum of $\log B_i$ is

$$\sum_{i=1}^N \log B_i(\theta) = -N \cdot U_i(\theta^0) - \Big(\sum_{i=1}^N \nabla U_i(\theta^0)\Big)^\intercal(\theta - \theta^0) - \frac{1}{2} \cdot \sum_{i=1}^N \bar{U}_{2,i} \cdot \left\|\theta - \theta^0\right\|_1^2.$$

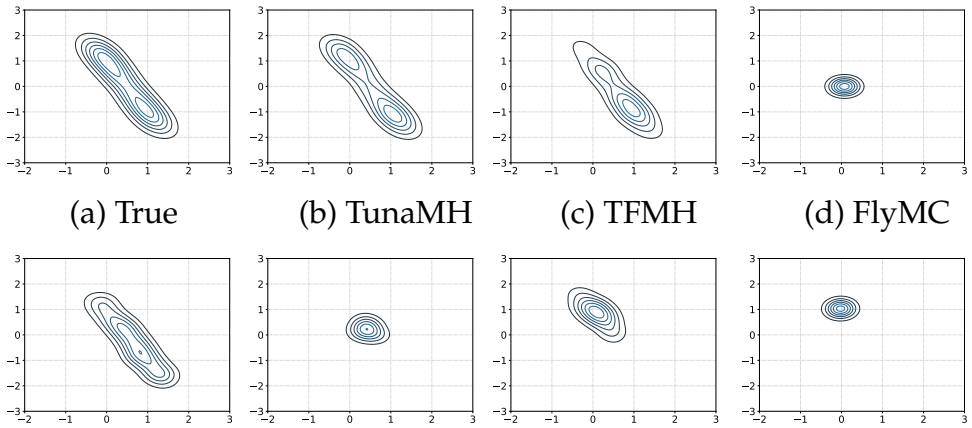
We set θ^0 to be 0 and the MAP solution in standard and MAP-tuned FlyMC respectively.

We tune the stepsize of each method to reach the acceptance rate 60% and the value of stepsize is summarized in Table D.1 and D.2. We set $\chi = 10^{-4}$ in TunaMH and $q_{d\to b} = 0.01$ in FlyMC and FlyMC-MAP. We compute the symmetric KL between the run-average density estimate and the true distribution. Since this is a two-dimensional problem, we are able to visualize the density estimate. As shown in Figure D.2, we plot the density estimate after running the method for 1 second. It is clear to see that the density estimate of TunaMH is close to the truth whereas all other methods are unable to provide accurate density estimate given the time budget.

D.10.4 Logistic Regression on MNIST

MNIST with only 7s and 9s images contains 12214 training data and 2037 test data. Let h be the sigmoid function. Let the label $y_i \in \{0, 1\}$, then the model in logistic regression (LR) is

$$p(y_i = 1) = h(\theta^\intercal x_i) = \frac{1}{1 + \exp(-\theta^\intercal x_i)}.$$



(a) True (b) TunaMH (c) TFMH (d) FlyMC

(e) PoissonMH (f) MH (g) SMH-1 (h) FlyMC-MAP

Figure D.2: Visualization of the density estimate after 1 second.

It follows that

$$U_i(\theta) = -y_i \log h\left(\theta^\intercal x_i\right) - (1 - y_i) \log h\left(-\theta^\intercal x_i\right).$$

It is easy to see that

$$\left|\partial_j U_i\right| = \left|(h(\theta^\intercal x_i) - y_i) x_{ij}\right| \leq 1 \cdot \left|x_{ij}\right|.$$

Thus we can set $M(\theta, \theta')$ to be $\|\theta - \theta'\|_2$ and c_i to be $\|x_i\|_2$. We use this bound for TunaMH, TFMH and SMH. For FlyMC, we use the same bound on logistic regression as in the FlyMC paper [390].

We set the target acceptance rate to be 60% and the resulted stepsize is reported in Table D.1. We set $q_{d\to b}$ to be 0.1 following Maclaurin et al. [390].

APPENDIX E

APPENDIX FOR CD-GRAB

Term	Explanation
GraB	Centralized online Gradient Balancing algorithm (Original centralized algorithm developed in GraB [384].
CD-GraB	Coordinated and distributed online Gradient Balancing algorithm. The algorithm that is our main contribution.
ID-GraB	Independent, distributed online gradient balancing. Implemented for our ablation study. One version uses the original GraB's online Balance (ID-GraB (Bal)), and one implements our online PairBalance (ID-GraB (PairBal)).
RR	Random reshuffling algorithm. We use this to refer to its centralized variant.
D-RR	Distributed random reshuffling algorithm.
SO	Shuffle Once algorithm.
$\boldsymbol{x}$	Data-example vector; we do not use this in the math in the main paper, but do refer to examples in our schematic description for PairBalance ordering in Figure 6.1.
$\boldsymbol{z}$	Vector (for illustration under the herding context). For GraB and CD-GraB, these are gradients.
$\bar{\boldsymbol{z}}$	The average vector (for illustration under the herding context).
$\boldsymbol{z}_j$	The j-th component of a vector (for illustration under the herding context).
$\boldsymbol{z}_{i,j}$	The j-th component of the gradient on worker i (for illustration under our parallel herding framework).
$\boldsymbol{w}$	Parameters / model-weights vector.
f	Loss function.
$\nabla f(\boldsymbol{w})$	Global loss gradient.
$\nabla f^i(\boldsymbol{w})$	Local i-th worker's loss gradient.
$\nabla f^i(\boldsymbol{w}; j)$	Local i-th worker's, j-th example's loss gradient.
π	A permutation; we study permutation-based example orderings.
T	Number of epochs.
t	Index for iterating over T epochs.
m	Number of workers (in this paper, workers are processes, potentially on different GPUs but on the same node). $m = 1$ in the centralized setting.
i	Index for iterating over m workers .
n	Number of training-data examples per worker; equivalent to $\frac{N}{m}$.
N	Number of total training-data examples. $N = n$ in the centralized setting.
j	Index for iterating over examples.
$\boldsymbol{g}$	Gradient, taken with respect to the model weights $\boldsymbol{w}$ and data examples $\boldsymbol{x}$.
$\boldsymbol{g}_j^i$	Gradient associated with the j-th data example $\boldsymbol{x}$ on worker i.
s	A sign, either +1 or −1; related to the signed herding problem.
s_j^i	A sign, either +1 or −1, computed according to the j-th example gradient $\boldsymbol{g}_j^i$ for worker i; to be associated with the example $\boldsymbol{x}_j$ when determining a permutation ordering using Algorithm 5.

E.1 Additional Details on the CD-GraB Algorithm and online PairBalance

In this Appendix, we provide more details on related work and our contributions. To start, we give a unified description of our online CD-GraB algorithm with prior work on herding, vector balancing, and kernel thinning (Appendix E.1.1), some more details on Alweiss et al. [20] that we elide in the main paper due to space constraints (Appendix E.1.2), conceptual details on implementing CD-GraB with a parameter server (Appendix E.1.3), and implementing our improved balancing algorithm (online PairBalance) in a centralized fashion to get additional improvements for GraB (Appendix E.1.4).

E.1.1 Distinguishing our contributions

We summarize our contributions in relation to prior work in a concise format. This kind of presentation would not be easily understandable without the appropriate background and context that we provide in the paper. This is why present it here, in the Appendix, so that (ideally) this is seen by the reader after finishing the main paper.

We emphasize that it is prior work that:

- Formulates the herding objective and solves it with vector balancing [261, 620] (Algorithm 5).
- Leverages ideas from herding and vector balancing (above) in an optimization setting to do permutation-based example ordering [384].

- Observes and proves that it is possible to solve the herding objective in $\tilde{O}(1)$ by only examining differences on pairs of examples (the overarching idea of PairBalance [182], which relies on the online RandomizedBalance subroutine [20]; see Algorithm 6).

Our contributions are to bring together all of this prior work in a novel way. We

- Translate the herding and balancing framework to the parallel setting via defining a parallel herding objective (6.8).
- Leverage prior work on herding in an optimization setting [384] so that we can do parallel herding in an optimization setting (Section 6.3).
- Execute *online* pair balancing on a server (Algorithm 6 on a running sum, Figure 6.1), i.e., do pair balancing in a streaming and asynchronous (rather than blocking) fashion from gradient vectors produced on distributed workers (Algorithm 6.2), on the flattened sequenced of paired-difference gradients (Section 6.3.2); this leads to an improvement over GraB, which relies on a stale mean.

E.1.2 More details on RandomizedBalance from Alweiss et al. [20]

In the subroutine for RandomizedBalance in Algorithm 6, we elide details about how the probability p is computed exactly as in Alweiss et al.[20]. We provide a more complete specification in Algorithm 11 written in terms of a single input

vector (which, for us, is the vector containing the difference between adjacent gradients). Note that the difference here is in the use of a required parameter, constant upper bound w, which is used to compute the probability p. For clarity of presentation in the subroutine in Algorithm 6, we have set $w = 1$. Alweiss et al. [20] sets this threshold differently, which we still elide for simplicity.

Algorithm 11 Probabilistic Balancing with Logarithm Bound [Alweiss et al. [20]]

require: parameter w, used to compute probability
input: current running sum $\boldsymbol{r}$ vector, vector z_{diff}
if $|\langle \boldsymbol{r}, z_{\text{diff}} \rangle| > w$ or $\|\boldsymbol{r}\|_\infty > w$ **then**
    Fail
end if
compute: $p \leftarrow \frac{1}{2} - \frac{\langle \boldsymbol{r}, z_{\text{diff}} \rangle}{2w}$
compute: $s \leftarrow +1$ with probability p;
    $s \leftarrow -1$ with probability $1 - p$
update: $\boldsymbol{r} \leftarrow \boldsymbol{r} + s z_{\text{diff}}$
return: s, $\boldsymbol{r}$

In practice, we actually do not use RandomizedBalance in our online PairBalance. We use the deterministic, greedy-ordering algorithm from the original Lu et al. [384, Algorithm 5] paper:

Algorithm 12 Balancing without normalization [Lu et al. [384]]

input: current running sum $\boldsymbol{r}$ vector, vector z_{diff}
if $\|\boldsymbol{r} + z_{\text{diff}}\| < \|\boldsymbol{r} - z_{\text{diff}}\|$ **then** $s \leftarrow +1$ **else** $s \leftarrow -1$
update: $\boldsymbol{r} \leftarrow \boldsymbol{r} + s z_{\text{diff}}$
return: s, $\boldsymbol{r}$

Note that, unlike Alweiss et al. [20] (Algorithm 11), Algorithm 12 from Lu et al. [384] cannot end up in a failure state.

In Alweiss et al. [20], Theorem 1.1 proves the $\tilde{O}(1)$ probabilistic bound for Algorithm 11 (See Theorem 5) for a restatement of this result in terms of our work). Corollary 7 of Dwivedi and Mackey [182] re-proves this result (which they mislabel as Alweiss et al.[20], Theorem 1.2, see Dwivedi and Mackey [182,

Appendix R, p. 69]). They improve the constants and have a less conservative setting of the thresholds w. The proof is also very short and elegant, by relying on their Theorem 3.

E.1.3 Implementing CD-GraB with a parameter server

For our implementation of CD-GraB, we use a parameter server architecture [370]. For our purposes, this just entails computing the average gradient (used to update the model on all workers) on the server side. That is, the server (other than determining the ordering for the next epoch) also has the function of aggregating gradient information (in this case, a simple mean) to send back to the workers.

We have the server compute the average j-th gradient for illustrative purposes. We could, instead, implement the computation the average gradient as an all-reduce operation, in which each worker broadcasts their gradients to all other workers, so that they can each locally compute the average gradient to update their local models. We implement CD-GraB using a parameter server pattern to show that this is a plausible architecture to use with our coordinated and distributed example ordering algorithm. We could also implement a full parameter server system, for which the server also coordinates global model updates.

If we kept everything in our implementation the same and switched to all-reduce, then we would no longer be following a parameter server paradigm. In this case, the server would just function to determine example orders. It is this kind of paradigm that suggests the abstraction of an *order server*, which we

mention briefly in Section 6.6: A server whose sole responsibility is coordinating worker information to determine example ordering.

In future work, we intend to explore a host of architectural possibilities — of building a full system that incorporates both traditional parameter server aspects with our new abstraction of an order server. For example, we could have parameter servers and order servers work in tandem in a distributed system to perform model training. To move beyond the single-node implementation we present in this paper, we intend to investigate the benefits and trade-offs associated with such design decisions in an actual implemented system.

E.1.4 Centralized online PairBalance

In Section 6.3.3, we provide a schematic diagram of how online PairBalance works for a distributed implementation using a parameter server (Figure 6.1). We also claim in Section 6.3 that online PairBalance can be applied to the original centralized GraB algorithm for improved empirical performance. We provide a schematic here, in Figure E.1 (analogous to Figure 6.1), for online PairBalance for centralized GraB.

We also provide empirical results comparing GraB's Balance routine to the online PairBalance routine that we instead use in this work. We observe that both PairBalance and Balance would have similar convergence rates under centralized settings, and both outperform RR.

This experiment justifies the uses of PairBalance even in centralized learning settings. PairBalance theoretically tolerates higher learning rates and, as we

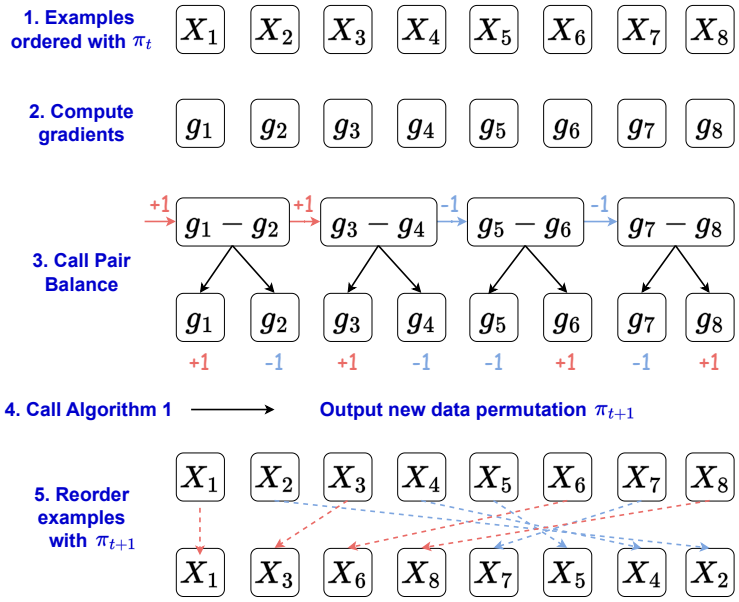


Figure E.1: Schematic representation of online PairBalance for centralized GraB.

will justify in Appendix E.3.1, is more memory-efficient than Balance. In short, PairBalance an excellent substitute for Balance when running GraB.

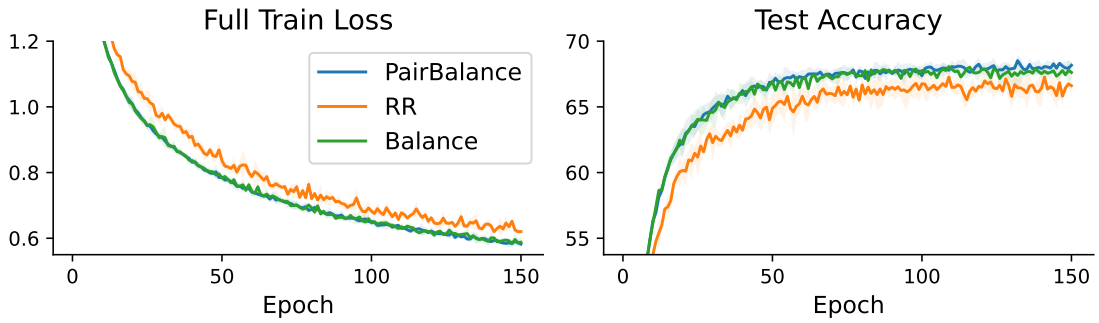


Figure E.2: Convergence for centralized online PairBalance on LeNet on CIFAR-10. We use the identical set of hyperparameters (α = 1e-3, weight decay = 1e-2, momentum = 0.9, B = 64) as in the scaling experiments as in Figure 6.4.

E.2 Proof Results

We present supporting results, which we use to prove the main results presented in Section 6.4. First, we show how to analyze the parallel herding bound in terms of a single step over the server-side PairBalance algorithm (Appendix E.2.1). We then include some additional observations/notation (Appendix E.2.2), which we use in the remaining intermediate results. We prove some intermediate results about how much the loss can change over the course of one epoch, assuming smoothness (Appendix E.2.3) and bounded gradient variance and heterogeneity (Appendix E.2.4). We combine these results to get one more intermediate result about the maximum the loss can change on average over many epochs (Appendix E.2.5), which we then use altogether to prove the two theorems that we present in the main paper (Appendix E.2.6).

E.2.1 Analyzing the parallel herding bound

In the main paper, we cover how CD-GraB runs on both the worker- and server-side. In this section, we dive deeper into the example-ordering part of CD-GraB, and demonstrate in theory how server-side online PairBalance reduces the parallel herding bound (6.8), as formulated in Section 6.3. We conclude this section by presenting Lemma 3, which shows server-side PairBalance is able to iteratively reduce the parallel herding bound.

To begin, we formalize our illustration over a group of vectors (since vector balancing, including PairBalance, does not inherently involve an optimization context until we use it in our online setting on gradients). Without loss of gen-

Algorithm 13 Server-side PairBalance over a set of vectors (one step)

require: m workers, $n \coloneqq \frac{N}{m}$ vectors per worker
input: initial permutations for all the workers $\{\pi_i\}_{i=1}^m$
initialize: new permutations for all the workers $\{\pi_i'\}_{i=1}^m$
initialize: running partial sum $\boldsymbol{h} = \mathbf{0}$
initialize: new indices front (left) pointer $\{l_i = 1\}_{i=1}^m$
initialize: new indices back (right) pointer $\{r_i = 1\}_{i=1}^m$
for example $j \coloneqq 1 \dots n$ **do**
  for worker $i \coloneqq 1 \dots m$ **do**
    if $j \bmod 2 = 0$ **then** ▷ If at an even index, i.e., can examine a full pair of examples
      $\boldsymbol{h},\ s_{j-1}^i,\ s_j^i \leftarrow \mathsf{PairBalance}(\boldsymbol{h},\ \boldsymbol{z}_{j-1}^i, \boldsymbol{z}_j^i)$
      if $s_{j-1}^i = +1$ **then**
        $\pi_i'(l_i) = j - 1; \quad l_i = l_i + 1$ ▷ Append first in pair to the front/left
        $\pi_i'(r_i) = j; \quad r_i = r_i - 1$ ▷ Append second in pair to the right/back
      else
        $\pi_i'(l_i) = j; \quad l_i = l_i + 1$ ▷ Append second in pair to the left/front
        $\pi_i'(r_i) = j - 1; \quad r_i = r_i - 1$ ▷ Append first in pair to the right/back
      end if
    end if
  end for
end for
output: new permutations for all m workers $\{\pi_i'\}_{i=1}^m$

Figure E.3: One-step PairBalance algorithm on the server side to solve the parallel herding problem (6.8). This algorithm can be seen as a prototype for Algorithms 7 and 8, without the optimization context.

erality, we assume that the N examples are divided evenly among the m workers and that n is even. That is, we consider that we are given a set of vectors $\boldsymbol{z}_{i,j} \in \mathbb{R}^d$ for $i \in [m]$ and $j \in [n]$ evenly located on m workers (i.e., $n = \frac{N}{m}$), where $\boldsymbol{z}_{i,j}$ denotes the j-th vector located on the i-th worker. Now denote π_i as the original permutation of the vectors on worker i. Consider running Algorithm 13 on the server side over these N vectors.

It follows, in the following Lemma 3, that we can get the parallel herding bound with the output permutations $\{\pi'_i\}_{i=1}^m$ from Algorithm 13:

Lemma 3. *Suppose that we have a set of vectors $z_{i,j} \in \mathbb{R}^d$ for all $i, i' \in [m]$ and for all $j, j' \in [n]$ that satisfies*

$$\left\| \sum_{i=1}^{m} \sum_{j=1}^{n} z_{i,j} \right\|_\infty \leq c_1 \qquad \textit{and} \qquad \left\| z_{i',j'} - \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} z_{i,j} \right\|_\infty \leq c_2$$

for some constants $c_1 > 0$ and $c_2 > 0$. If we run Algorithm 13 over these vectors, then, for any $\delta > 0$, it holds with probability at least $1 - \delta$ that

$$\max_{l \in [n]} \left\| \sum_{i=1}^{m} \sum_{j=1}^{l} z_{i,\pi'_i(j)} \right\|_\infty \leq \frac{1}{2} \max_{l \in [n]} \left\| \sum_{i=1}^{m} \sum_{j=1}^{l} z_{i,\pi_i(j)} \right\|_\infty + c_1 + \tilde{A} c_2,$$

where $\tilde{A}$ comes from Theorem 5.

Lemma 3 shows that PairBalance reduces the parallel herding objective (6.8) towards a constant (invariant to n) at each step. This implies that, if we repeatedly call PairBalance on a given permutation, it will return a permutation that guarantees the parallel herding bound to be $\tilde{O}(1)$.

Proof. We prove this lemma by defining the following auxiliary sequence of pair differences, as in Section 6.3.2

$$y_{n \cdot (k-1)+i} = z_{i,\pi_i(2k-1)} - z_{i,\pi_i(2k)}, \quad \forall k \in [n/2],$$

which we also can refer to as $\{y_j\}_{j=1}^{mn/2}$.

We also leverage Theorem 5, which we reprint below for clarity of presentation:

Theorem 1 (Corollary 7, Dwivedi and Mackey [182]). *Consider any vectors $\{z_j\}_{j=1}^N$ ($z_j \in \mathbb{R}^d$) with $\|z_j\|_2 \leq 1$ supplied as input to the* RandomizedBalance *subroutine in*

Algorithm 6. Then for any $\delta > 0$, with probability at least $1 - \delta$, RandomizedBalance *outputs a sequence of signs $\{s_j\}_{j=1}^N \in \{-1, 1\}$ that satisfy $\max_{k\in[N]} \left\|\sum_{j=1}^k s_j \boldsymbol{z}_j\right\|_\infty \leq \tilde{A}$, where $\tilde{A} = \sqrt{2\log\left(\frac{4d}{\delta}\right)\log\left(\frac{4N}{\delta}\right)} = \tilde{O}(1)$.*

Note that the reordering part of Algorithm 13 (line 8) gives a sequence of signs $\{s_j\}_{j=1}^{mn/2}$. Therefore, by Theorem 5, the sequence $\{\boldsymbol{y}_j\}_{j=1}^{mn/2}$ satisfies

$$\max_{P\in[mn/2]} \left\|\sum_{p=1}^{P} s_p \boldsymbol{y}_p\right\|_\infty \leq 2\tilde{A}c_2, \tag{E.1}$$

since (based on what is given in Lemma 3)

$$\left\|\boldsymbol{y}_{n(k-1)+i}\right\|_\infty \leq \left\|\boldsymbol{z}_{i,\pi_{t,i}(2k-1)} - \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\boldsymbol{z}_{i,j}\right\|_\infty + \left\|\boldsymbol{z}_{i,\pi_{t,i}(2k)} - \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\boldsymbol{z}_{i,j}\right\|_\infty \leq 2c_2.$$

Note that, if $s_{i,k}$ is the sign associated with $\boldsymbol{y}_{n(k-1)+i}$, then $\boldsymbol{z}_{i,\pi_{t,i}(2k-1)}$ and $\boldsymbol{z}_{i,\pi_{t,i}(2k)}$ will receive opposite signs $s_{i,k}$ and $-s_{i,k}$, respectively.

We denote $\boldsymbol{x}_{i,k}^+$ to be the example that receives sign $s_{i,k} = +1$ and $\boldsymbol{x}_{i,k}^-$ to be the example that receives sign $s_{i,k} = -1$.

That is, if $s_{i,k} = +1$, then $\boldsymbol{x}_{i,k}^+ = \boldsymbol{z}_{i,\pi_i(2k-1)}$, otherwise, if $s_{i,k} = -1$, then $\boldsymbol{x}_{i,k}^+ = \boldsymbol{z}_{i,\pi_i(2k)}$; and, $\boldsymbol{x}_{i,k}^-$ is the other term of the pair $\{\boldsymbol{z}_{i,\pi_i(2k-1)}, \boldsymbol{z}_{i,\,\pi_i(2k)}\}$.

Now, for $K \in [\frac{n}{2}]$, let

$$\kappa_{i,K} = \sum_{k=1}^{K}(\boldsymbol{z}_{i,\pi_i(2k-1)} + \boldsymbol{z}_{i,\pi_i(2k)}) \text{ and}$$
$$\upsilon_{i,K} = \sum_{k=1}^{K}(s_{i,k}\boldsymbol{z}_{i,\pi_i(2k-1)} - s_{i,k}\boldsymbol{z}_{i,\pi_i(2k)}).$$

Then

$$\sum_{k=1}^{K}\boldsymbol{x}_{i,k}^+ = \frac{1}{2}(\kappa_{i,K} + \upsilon_{i,K}) \quad \text{and} \quad \sum_{k=1}^{K}\boldsymbol{x}_{i,k}^- = \frac{1}{2}(\kappa_{i,K} - \upsilon_{i,K}).$$

Now, observe that

$$\sum_{i=1}^{m} \kappa_{i,K} = \sum_{j=1}^{2K} \sum_{i=1}^{m} \boldsymbol{z}_{i,\pi_i(j)} \quad \text{and} \quad \sum_{i=1}^{m} \upsilon_{i,K} = \sum_{p=1}^{mK} s_p \boldsymbol{y}_p.$$

Therefore,

$$\begin{aligned}
\max_{K\in[n/2]} \left\| \sum_{k=1}^{K} \sum_{i=1}^{m} \boldsymbol{x}_{i,k}^{+} \right\|_\infty &\le \frac{1}{2} \left(\max_{K\in[n/2]} \left\| \sum_{i=1}^{m} \kappa_{K,i} \right\|_\infty + \max_{K\in[n/2]} \left\| \sum_{i=1}^{m} \upsilon_{K,i} \right\|_\infty \right) \\
&\le \frac{1}{2} \max_{K\in[n/2]} \left\| \sum_{j=1}^{2K} \sum_{i=1}^{m} \boldsymbol{z}_{i,j} \right\|_\infty + \tilde{A}c_2 \quad \text{By substituting above and (E.1)} \\
&\le \frac{1}{2} \max_{k\in[n]} \left\| \sum_{j=1}^{k} \sum_{i=1}^{m} \boldsymbol{z}_{i,j} \right\|_\infty + \tilde{A}c_2.
\end{aligned}$$

And similarly,

$$\begin{aligned}
\max_{K\in[n/2]} \left\| \sum_{k=1}^{K} \sum_{i=1}^{m} \boldsymbol{x}_{i,k}^{-} \right\| &\le \frac{1}{2} \left(\max_{K\in[n/2]} \left\| \sum_{i=1}^{m} \kappa_{K,i} \right\|_\infty + \max_{K\in[n/2]} \left\| \sum_{i=1}^{m} \upsilon_{K,i} \right\|_\infty \right) \\
&\le \frac{1}{2} \max_{k\in[n]} \left\| \sum_{j=1}^{k} \sum_{i=1}^{m} \boldsymbol{z}_{i,j} \right\|_\infty + \tilde{A}c_2.
\end{aligned}$$

Applying the new permutation $\pi_i'(j)$ on the vectors $\boldsymbol{z}_{i,\pi_i(j)}$, we get for each $i \in [m]$ the permuted sequence

$$\boldsymbol{x}_{i,1}^{+}, \ldots, \boldsymbol{x}_{i,n/2}^{+}, \boldsymbol{x}_{i,n/2}^{-}, \ldots, \boldsymbol{x}_{i,1}^{-}.$$

Thus, we need to bound the herding objective of the sequence

$$\sum_{i=1}^{m} \boldsymbol{x}_{i,1}^{+}, \ldots, \sum_{i=1}^{m} \boldsymbol{x}_{i,n/2}^{+}, \sum_{i=1}^{m} \boldsymbol{x}_{i,n/2}^{-}, \ldots, \sum_{i=1}^{m} \boldsymbol{x}_{i,1}^{-}.$$

If the partial sums above peak at $t_0 \le n/2$, then we can bound the parallel herding objective as

$$\left\| \sum_{k=1}^{t_0} \sum_{i=1}^{m} \boldsymbol{x}_{i,k}^{+} \right\|_\infty = \max_{K\in[n/2]} \left\| \sum_{k=1}^{K} \sum_{i=1}^{m} \boldsymbol{x}_{i,k}^{+} \right\|_\infty \le \frac{1}{2} \max_{k\in[n]} \left\| \sum_{j=1}^{k} \sum_{i=1}^{m} \boldsymbol{z}_{i,j} \right\|_\infty + \tilde{A}c_2;$$

otherwise, we can bound the parallel herding objective as

$$\left\|\sum_{j=1}^{n}\sum_{i=1}^{m}\boldsymbol{z}_{i,j}-\sum_{k=1}^{m-t_0}\sum_{i=1}^{m}\boldsymbol{x}_{i,k}^{-}\right\|_\infty \le \left\|\sum_{j=1}^{n}\sum_{i=1}^{m}\boldsymbol{z}_{i,j}\right\|_\infty + \left\|\sum_{k=1}^{m-t_0}\sum_{i=1}^{m}\boldsymbol{x}_{i,k}^{-}\right\|_\infty$$
$$\le c_1 + \frac{1}{2}\max_{t\in[n]}\left\|\sum_{j=1}^{t}\sum_{i=1}^{m}\boldsymbol{z}_{i,j}\right\|_\infty + \tilde{A}c_2,$$

since in Algorithm 5 the list of vectors with negative signs is reversed before concatenated.

The claim follows. □

E.2.2 Notation and observations

We begin with three notes that we will use throughout the intermediate results we present in this section. We will use the lemmas presented here to prove our main results: Theorems 6 and 7 in Appendix E.2.6.

1. **A single t-th update.** First, recall that one t-th step of the parameter update can be written as

$$\boldsymbol{w}_t^{j+1} = \boldsymbol{w}_t^{j} - \frac{\alpha}{m}\sum_{i=1}^{m}\nabla f^i(\boldsymbol{w}_t^{j};\ \pi_{t,i}(j)), \quad \forall\, j\in[n]$$

   We will use the convention $\boldsymbol{w}_{t+1} \triangleq \boldsymbol{w}_{t+1}^{1} \triangleq \boldsymbol{w}_t^{n+1}$.

2. **The maximum amount a parameter can change over an epoch.** The key quantity in our proof is Δ_t, which is the maximum amount that a parameter in $\boldsymbol{w}$ can change in epoch t. That is,

$$\Delta_t \triangleq \max_{k\in[n]} \left\|\boldsymbol{w}_t^{k+1} - \boldsymbol{w}_t\right\|_\infty$$
$$= \frac{\alpha}{m} \max_{k\in[n]} \left\| \sum_{j=1}^{k} \sum_{i=1}^{m} \nabla f^i(\boldsymbol{w}_t^j; \pi_{t,i}(j)) \right\|_\infty . \tag{E.2}$$

Following this definition of Δ_t, we note that the maximum amount that a parameter in $\boldsymbol{w}$ can change over two different epochs is $2\Delta_t$. That is, we observe

$$\left\|\boldsymbol{w}_t^j - \boldsymbol{w}_t^k\right\|_\infty \leq \left\|\boldsymbol{w}_t^j - \boldsymbol{w}_t\right\|_\infty + \left\|\boldsymbol{w}_t^k - \boldsymbol{w}_t\right\|_\infty \leq 2\Delta_t$$
$$\left\|\boldsymbol{w}_{t+1}^j - \boldsymbol{w}_t^k\right\|_\infty \leq \left\|\boldsymbol{w}_{t+1}^j - \boldsymbol{w}_{t+1}\right\|_\infty + \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|_\infty + \left\|\boldsymbol{w}_t^k - \boldsymbol{w}_t\right\|_\infty \leq \Delta_{t+1} + 2\Delta_t, \quad \forall j,k \in [n].$$

We make repeated use of this relation in the results that follow, which we typically will use in combination with the Lipschitz assumption to bound gradients of the same loss function but with different parameters.

3. **Bounding loss at epoch t.** We will denote $F_t = f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)$ where $\boldsymbol{w}^*$ is the minimizer of f which we assume to be bounded from below.

E.2.3 Assuming $L_{2,\infty}$-smoothness: results on the amount the loss can change over one epoch)

We will next prove an intermediate result regarding that bounds the loss f at epoch $t+1$ in relation to the loss at the prior epoch t (Lemma 4). That is, we prove results about how much the loss with respect to the parameters can change over the course of one epoch.

Lemma 4. *If the loss f is $L_{2,\infty}$-smooth and the learning rate $\alpha \leq \frac{1}{nL_{2,\infty}}$, then*

$$f(\boldsymbol{w}_{t+1}) \leq f(\boldsymbol{w}_t) + \frac{\alpha n L_{2,\infty}^2}{2}\Delta_t^2 - \frac{\alpha n}{2}\|\nabla f(\boldsymbol{w}_t)\|_2^2. \tag{E.3}$$

Proof. We begin with the definition of $L_{2,\infty}$-smoothness, with respect to loss f:

$$f(\boldsymbol{w}_{t+1}) \leq f(\boldsymbol{w}_t) + \nabla f(\boldsymbol{w}_t)^\top(\boldsymbol{w}_{t+1} - \boldsymbol{w}_t) + \frac{L_{2,\infty}}{2}\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|_2^2$$

Also observe that

$$\begin{aligned} -\nabla f(\boldsymbol{w}_t)^\top(\boldsymbol{w}_t - \boldsymbol{w}_{t+1}) &= -\frac{\alpha n}{2} 2\nabla f(\boldsymbol{w}_t)^\top \left(\frac{\boldsymbol{w}_t - \boldsymbol{w}_{t+1}}{\alpha n}\right) \\ &= \frac{\alpha n}{2}\left(\left\|\nabla f(\boldsymbol{w}_t) - \frac{(\boldsymbol{w}_t - \boldsymbol{w}_{t+1})}{\alpha n}\right\|_2^2 - \|\nabla f(\boldsymbol{w}_t)\|_2^2 - \left\|\frac{(\boldsymbol{w}_t - \boldsymbol{w}_{t+1})}{\alpha n}\right\|_2^2\right). \end{aligned} \tag{E.4}$$

Combining the above — i.e., the definition of $L_{2,\infty}$-smoothness with (E.4) — we get

$$f(\boldsymbol{w}_{t+1}) \leq f(\boldsymbol{w}_t) + \frac{\alpha n}{2}\left\|\nabla f(\boldsymbol{w}_t) - \frac{(\boldsymbol{w}_t - \boldsymbol{w}_{t+1})}{\alpha n}\right\|_2^2 - \frac{\alpha n}{2}\|\nabla f(\boldsymbol{w}_t)\|_2^2 \qquad + \frac{\alpha n L_{2,\infty} - 1}{2\alpha n}\|\boldsymbol{w}_t - \boldsymbol{w}_{t+1}\|_2^2.$$

The last term on the right-hand side is ≤ 0 by the assumption that the learning rate $\alpha \leq \frac{1}{nL_{2,\infty}}$. Therefore,

$$f(\boldsymbol{w}_{t+1}) \leq f(\boldsymbol{w}_t) + \frac{\alpha n}{2}\left\|\nabla f(\boldsymbol{w}_t) - \frac{(\boldsymbol{w}_t - \boldsymbol{w}_{t+1})}{\alpha n}\right\|_2^2 - \frac{\alpha n}{2}\|\nabla f(\boldsymbol{w}_t)\|_2^2. \tag{E.5}$$

We next bound the second term on the right-hand side by Δ_t (E.2):

$$\begin{aligned}\left\|\nabla f(\boldsymbol{w}_t) - \frac{(\boldsymbol{w}_t - \boldsymbol{w}_{t+1})}{\alpha n}\right\|_2^2 &= \left\|\frac{1}{mn}\sum_{j=1}^{m}\sum_{i=1}^{n}\nabla f^i(\boldsymbol{w}_t, \pi_t(j)) - \frac{1}{mn}\sum_{j=1}^{m}\sum_{i=1}^{n}\nabla f^i(\boldsymbol{w}_t^j; \pi_t(j))\right\|_2^2 \\ &\leq \frac{1}{mn}\sum_{j=1}^{m}\sum_{i=1}^{n}\left\|\nabla f^i(\boldsymbol{w}_t, \pi_t(j)) - \nabla f^i(\boldsymbol{w}_t^j; \pi_t(j))\right\|_2^2 \\ &\leq \frac{L_{2,\infty}^2}{mn}\sum_{j=1}^{m}\sum_{i=1}^{n}\left\|\boldsymbol{w}_t^j - \boldsymbol{w}_t\right\|_\infty^2,\end{aligned}$$

where we have used $L_{2,\infty}$-smoothness (Assumption 4) in the last inequality. Substituting Δ_t, we get

$$\frac{L_{2,\infty}^2}{mn}\sum_{j=1}^{m}\sum_{i=1}^{n}\left\|\boldsymbol{w}_t^j - \boldsymbol{w}_t\right\|_\infty^2 \quad \leq \quad L_{2,\infty}^2\Delta_t^2.$$

Plugging the above into (E.5), we get

$$f(\boldsymbol{w}_{t+1}) \leq f(\boldsymbol{w}_t) + \frac{\alpha n L_{2,\infty}^2}{2}\Delta_t^2 - \frac{\alpha n}{2}\|\nabla f(\boldsymbol{w}_t)\|_2^2,$$

yielding the claim. □

We next build slightly on Lemma 4 to make two additional observations. First:

Lemma 5. *If the loss f is $L_{2,\infty}$-smooth and the learning rate $\alpha \leq \frac{1}{nL_{2,\infty}}$, then*

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\boldsymbol{w}_t)\|_2^2 \leq \frac{2F_1}{\alpha n T} + \frac{L_{2,\infty}^2}{T}\sum_{t=1}^{T}\Delta_t^2,$$

where F_1 comes from Theorem 6.

Proof. Using Lemma 4 and Jensen's inequality, we average (E.3) over $t \in [T]$ and match terms, yielding

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\boldsymbol{w}_t)\|_2^2 \leq \frac{2(f(\boldsymbol{w}_1) - f(\boldsymbol{w}_{T+1}))}{\alpha n T} + \frac{L_{2,\infty}^2}{T}\sum_{t=1}^{T}\Delta_t^2.$$

Substituting F_1, we get

$$\leq \frac{2F_1}{\alpha n T} + \frac{L_{2,\infty}^2}{T} \sum_{t=1}^{T} \Delta_t^2,$$

yielding the claim. □

We next build on Lemma 4 by further assuming the P.L. assumption holds.

Lemma 6. *If the loss f is $L_{2,\infty}$-smooth, the learning rate $\alpha \leq \frac{1}{nL_{2,\infty}}$, and the P.L. assumption (Assumption 5) holds, then, for $\rho = 1 - \frac{\alpha n \mu}{2}$*

$$F_{T+1} \leq \rho^T F_1 + \frac{\alpha n L_{2,\infty}^2}{2} \sum_{t=1}^{T} \rho^{T-t} \left(\Delta_t^2 - \frac{1}{2L_{2,\infty}^2} \|\nabla f(\boldsymbol{w}_t)\|_2^2 \right).$$

Proof. From Lemma 4, we got (E.3), i.e.,

$$f(\boldsymbol{w}_{t+1}) \leq f(\boldsymbol{w}_t) + \frac{\alpha n L_{2,\infty}^2}{2} \Delta_t^2 - \frac{\alpha n}{2} \|\nabla f(\boldsymbol{w}_t)\|_2^2,$$

Applying the P.L. assumption (Assumption 5) to (E.3), we get

$$\begin{aligned} f(\boldsymbol{w}_{t+1}) &\leq f(\boldsymbol{w}_t) + \frac{\alpha n L_{2,\infty}^2}{2} \Delta_t^2 - \frac{\alpha n}{4} \|\nabla f(\boldsymbol{w}_t)\|_2^2 - \frac{\alpha n}{4} \|\nabla f(\boldsymbol{w}_t)\|_2^2 \\ &\leq f(\boldsymbol{w}_t) + \frac{\alpha n L_{2,\infty}^2}{2} \Delta_t^2 - \frac{\alpha n \mu}{2} (f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)) - \frac{\alpha n}{4} \|\nabla f(\boldsymbol{w}_t)\|_2^2. \end{aligned}$$

Subtracting f^* from both sides, we get

$$f(\boldsymbol{w}_{t+1}) - f^* \leq \left(1 - \frac{\alpha n \mu}{2}\right) (f(\boldsymbol{w}_t) - f^*) + \frac{\alpha n}{2} \left(L_{2,\infty}^2 \Delta_t^2 - \frac{1}{2} \|\nabla f(\boldsymbol{w}_t)\|_2^2 \right).$$

For $\rho = 1 - \frac{\alpha n \mu}{2}$, we then apply the above inequality recursively for $t \in [T]$, yielding the claim:

$$F_{T+1} \leq \rho^T F_1 + \frac{\alpha n L_{2,\infty}^2}{2} \sum_{t=1}^{T} \rho^{T-t} \left(\Delta_t^2 - \frac{1}{2L_{2,\infty}^2} \|\nabla f(\boldsymbol{w}_t)\|_2^2 \right).$$

□

E.2.4 Assuming bounded gradient variance and heterogeneity: results applying Algorithm 13

We next prove a result that builds on Lemma 3 and our one-step version of the server-side PairBalance algorithm (Algorithm 13).

We begin by introducing some additional notation. Namely, we will call π^{-1} the operation that, given an example, yields the index in the permutation for that example. For instance, $\pi_{t+1,i}(j)$ returns the example at the j-th index for the i-th worker's $t+1$ permutation. Let us denote that example τ. Then, $\pi_{t,i}^{-1}\pi_{t+1,i}(j)$ is equivalent to applying $\pi_{t,i}^{-1}$ to τ: it takes the example τ and returns τ's associated index in the i-th worker's epoch t's permutation (in this case, the prior epoch's permutation).

We will make use of this notation in the following Lemma.

Lemma 7. *Assume bounded gradient variance (Assumption 2), bounded gradient heterogeneity (Assumption 3), and $L_{2,\infty}$-smoothness (Assumption 4). For $t \in [T]$ and $\delta > 0$, if we apply Algorithm 13 to the gradients $\nabla f^i(\boldsymbol{w}_t^j; \pi_t^i(j))$ at epoch t to produce the next permutation $\pi_{t+1,i}$ for epoch $t+1$, then, with probability at least $1-\delta$,*

$$\Delta_{t+1} \quad \leq \quad \frac{1}{2}\Delta_t \quad + \quad \alpha L_{2,\infty}\left(4n + \frac{2\tilde{A}}{m}\right)\Delta_t \quad + \quad \alpha n L_{2,\infty}\Delta_{t+1} \quad + \quad \frac{\alpha(\varsigma+\sigma)\tilde{A}}{m} \quad + \quad \alpha n\|\nabla f(\boldsymbol{w}_{t+1})\|_2,$$

where $\tilde{A}$ comes from Theorem 5.

Proof. We start with the triangle inequality:

$$\left\|\sum_{j=1}^{k}\sum_{i=1}^{m}\nabla f^i(\boldsymbol{w}_{t+1}^j; \pi_{t+1,i}(j))\right\|_\infty \leq \left\|\sum_{j=1}^{k}\sum_{i=1}^{m}\nabla f^i(\boldsymbol{w}_t^{\pi_{t,i}^{-1}\pi_{t+1,i}(j)}; \pi_{t+1,i}(j))\right\|_\infty + \left\|\sum_{j=1}^{k}\sum_{i=1}^{m}\left(\nabla f^i(\boldsymbol{w}_{t+1,i}^j, \pi_{t+1,i}(j)) - \nabla f^i(\boldsymbol{w}_{t,i}^{\pi_{t,i}^{-1}\pi_{t+1,i}(j)}; \pi_{t+1,i}(j))\right)\right\|_\infty \tag{E.6}$$

We use Lemma 3 to bound the first term on the right-hand side of (E.6) from Lemma 4.

That is, let

$$z_{i,j} = \nabla f^i(\boldsymbol{w}_t^{\pi_{t,i}^{-1}(j)}; j),$$

so that

$$z_{i,\pi_{t+1,i}(j)} = \nabla f^i(\boldsymbol{w}_t^{\pi_{t,i}^{-1}\pi_{t+1,i}(j)}; \pi_{t+1,i}(j)).$$

The upper bounds for $\left\|z_{i,j} - \frac{1}{mn}\sum_{r,s} z_{r,s}\right\|_\infty$ and $\left\|\sum_{i,j} z_{i,j}\right\|_\infty$ are:

$$\left\|\nabla f^i(\boldsymbol{w}_t^j; \pi_{t,i}(j)) - \frac{1}{mn}\sum_{r=1}^{m}\sum_{s=1}^{n}\nabla f^s(\boldsymbol{w}_t^r; \pi_{t,s}(r))\right\|_\infty,$$

which are

$$\begin{aligned}\leq &\left\|\nabla f^i(\boldsymbol{w}_t^j; \pi_{t,i}(j)) - \frac{1}{mn}\sum_{r=1}^{m}\sum_{s=1}^{n}\nabla f^s(\boldsymbol{w}_t^j; \pi_{t,s}(r))\right\|_\infty + \\ &\left\|\frac{1}{mn}\sum_{r=1}^{m}\sum_{s=1}^{n}\nabla f^s(\boldsymbol{w}_t^j; \pi_{t,s}(r)) - \frac{1}{mn}\sum_{r=1}^{m}\sum_{s=1}^{n}\nabla f^s(\boldsymbol{w}_t^r; \pi_{t,s}(r))\right\|_\infty.\end{aligned}$$

We can rewrite the above to be

$$\begin{aligned}&\leq \left\|\nabla f^i(\boldsymbol{w}_t^j; \pi_{t,i}(j)) - \nabla f(\boldsymbol{w}_t^j)\right\|_\infty + \frac{L_{2,\infty}}{mn}\sum_{r=1}^{m} m \sum_{s=1}^{n}\left\|\boldsymbol{w}_t^j - \boldsymbol{w}_t^r\right\|_\infty \\ &\leq \varsigma + \sigma + 2L_{2,\infty}\Delta_t,\end{aligned}$$

by Assumptions 2, 3, and 4, and by the definition of Δ_t (E.2).

Now, observe that

$$\left\|\sum_{i=1}^{m}\sum_{j=1}^{n}\nabla f^i(\boldsymbol{w}_t^j; \pi_{t,i}(j))\right\|_\infty \leq \left\|\sum_{i=1}^{m}\sum_{j=1}^{n}\nabla f^i(\boldsymbol{w}_t^j; \pi_{t,i}(j)) - \sum_{i=1}^{m}\sum_{j=1}^{n}\nabla f^i(\boldsymbol{w}_{t+1}; \pi_{t,i}(j))\right\|_\infty + \left\|\sum_{i=1}^{m}\sum_{j=1}^{n}\nabla f^i(\boldsymbol{w}_{t+1}; \pi_{t,i}(j))\right\|_\infty.$$

By using the above, we can rewrite the right-hand side to be

$$\leq \sum_{i=1}^{m}\sum_{j=1}^{n} L_{2,\infty}\left\|\boldsymbol{w}_t^j - \boldsymbol{w}_{t+1}\right\|_\infty + mn\|\nabla f(\boldsymbol{w}_{t+1})\|_\infty$$

$$\leq 2mnL_{2,\infty}\Delta_t + mn\|\nabla f(\boldsymbol{w}_{t+1})\|_2.$$

Therefore, by Lemma 3,

$$\max_{k\in[n]}\left\|\sum_{j=1}^{k}\sum_{i=1}^{m}\nabla f^i(\boldsymbol{w}_t^{\pi_{t,i}^{-1}\pi_{t+1,i}(j)}, \pi_{t+1,i}(j))\right\|_\infty \leq \max_{k\in[n]}\left\|\sum_{j=1}^{k}\sum_{i=1}^{m}\nabla f^i(\boldsymbol{w}_t^j, \pi_{t,i}(j))\right\|_\infty$$
$$+2mnL_{2,\infty}\Delta_t + \|\nabla f(\boldsymbol{w}_{t+1})\|_2 + (\varsigma + \sigma + 2L_{2,\infty}\Delta_t)\tilde{A}.$$

The second term of the triangle inequality (E.6) can be bounded as

$$\left\|\sum_{j=1}^{k}\sum_{i=1}^{m}\left(\nabla f^i(\boldsymbol{w}_{t+1,i}^j; \pi_{t+1,i}(j)) - \nabla f^i(\boldsymbol{w}_{t,i}^{\pi_{t,i}^{-1}\pi_{t+1,i}(j)}; \pi_{t+1,i}(j))\right)\right\|_\infty,$$

which is

$$\leq \sum_{j=1}^{k}\sum_{i=1}^{m}\left\|\boldsymbol{w}_{t+1,i}^j - \boldsymbol{w}_{t,i}^{\pi_{t,i}^{-1}\pi_{t+1,i}(j)}\right\|_\infty$$

$$\leq mnL_{2,\infty}(\Delta_{t+1} + 2\Delta_t).$$

Substituting these bounds into the right-hand side of the triangle inequality (E.6), taking the max of both sides, and grouping terms, we get

$$\max_{k\in[n]}\left\|\sum_{j=1}^{k}\sum_{i=1}^{m}\nabla f^i(\boldsymbol{w}_{t+1,i}^j, \pi_{t+1,i}(j))\right\|_\infty \leq \frac{1}{2}\max_{k\in[n]}\left\|\sum_{j=1}^{k}\sum_{i=1}^{m}\nabla f^i(\boldsymbol{w}_t^j, \pi_{t,i}(j))\right\|_\infty$$
$$+ L_{2,\infty}(4mn + 2\tilde{A})\Delta_t + mnL_{2,\infty}\Delta_{t+1} + (\varsigma + \sigma)\tilde{A}$$
$$+ mn\|\nabla f(\boldsymbol{w}_{t+1})\|_2.$$

Multiplying both sides by $\frac{\alpha}{m}$ and using the definition of Δ_t (E.2), we get the claim. □

E.2.5 Combining the prior intermediate results: proofs over multiple steps

Lemma 8. *If the learning rate* $\alpha \leq \frac{1}{16L_{2,\infty}(2n+\tilde{A}/m)}$*, then*

$$\frac{1}{T}\sum_{t=1}^{T}\Delta_t^2 \leq \frac{21\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{m^2} + \frac{9\alpha^2n^2\sigma^2}{T} + 21\alpha^2n^2\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\boldsymbol{w}_t)\|_2^2.$$

Proof. First, we bound Δ_1^2.

We start with a series of triangle inequalities:

$$\begin{aligned}
\frac{\alpha}{m}\left\|\sum_{j=1}^{k}\sum_{i=1}^{m}\nabla f^i(\boldsymbol{w}_1^j, \pi_{1,i}(j))\right\|_\infty \leq& \frac{\alpha}{m}\left\|\sum_{j=1}^{k}\sum_{i=1}^{m}\nabla f^i(\boldsymbol{w}_1^j, \pi_{1,i}(j)) - \sum_{j=1}^{k}\sum_{i=1}^{m}\nabla f^i(\boldsymbol{w}_1, \pi_{1,i}(j))\right\|_\infty \\
&+ \frac{\alpha}{m}\left\|\sum_{j=1}^{k}\sum_{i=1}^{m}\Big(\nabla f^i(\boldsymbol{w}_1, \pi_{1,i}(j)) - \nabla f^i(\boldsymbol{w}_1)\Big)\right\|_\infty + \alpha k\|\nabla f(\boldsymbol{w}_1)\|_\infty \\
\leq& \frac{\alpha}{m}\sum_{j=1}^{k}\sum_{i=1}^{m}L_{2,\infty}\left\|\boldsymbol{w}_1^j - \boldsymbol{w}_1\right\|_\infty + \alpha k\sigma + \alpha k\|\nabla f(\boldsymbol{w}_1)\|_2.
\end{aligned}$$

We next take the max of both sides with respect to $k \in [n]$:

$$\begin{aligned}
\Delta_1 &\leq \alpha nL_{2,\infty}\Delta_1 + \alpha n\sigma + \alpha n\|\nabla f(\boldsymbol{w}_1)\|_2 \\
&\leq (1/32)\Delta_1 + \alpha n\sigma + \alpha n\|\nabla f(\boldsymbol{w}_1)\|_2 \qquad (\text{since } \alpha \leq \frac{1}{32nL_{2,\infty}}) \\
&\leq (32/31)\alpha n\sigma + (32/31)\alpha n\|\nabla f(\boldsymbol{w}_1)\|_2,
\end{aligned}$$

Squaring both sides:

$$\Delta_1^2 \leq 3\alpha^2n^2\sigma^2 + 3\alpha^2n^2\|\nabla f(\boldsymbol{w}_1)\|_2^2. \tag{E.7}$$

Now, we use Lemma 7 to get the relationship between Δ_{t+1} and Δ_t for $t \in [T]$.

Recall that

$$\Delta_{t+1} \leq \frac{1}{2}\Delta_t + \alpha L_{2,\infty}\left(4n + \frac{2\tilde{A}}{m}\right)\Delta_t + \alpha n L_{2,\infty}\Delta_{t+1} + \frac{\alpha(\varsigma+\sigma)\tilde{A}}{m} + \alpha n\|\nabla f(\boldsymbol{w}_{t+1})\|_2$$

Because $\alpha \leq \frac{1}{16L_{2,\infty}(2n+\tilde{A}/m)}$, we can rewrite the above as

$$\Delta_{t+1} \leq \frac{1}{2}\Delta_t + (1/8)\Delta_t + (1/32)\Delta_{t+1} + \frac{\alpha(\varsigma+\sigma)\tilde{A}}{m} + \alpha n\|\nabla f(\boldsymbol{w}_{t+1})\|_2.$$

Squaring both sides:

$$\begin{aligned}(31/32)^2\Delta_{t+1}^2 &\leq \frac{1}{2}\Delta_t^2 + 2\left((1/8)\Delta_t + \frac{\alpha(\varsigma+\sigma)\tilde{A}}{m} + \alpha n\|\nabla f(\boldsymbol{w}_{t+1})\|_2\right)^2 \\ &\leq \frac{1}{2}\Delta_t^2 + (6/8^2)\Delta_t^2 + \frac{6\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{m^2} + 6\alpha^2n^2\|\nabla f(\boldsymbol{w}_{t+1})\|_2^2,\end{aligned}$$

so that

$$\begin{aligned}\Delta_{t+1}^2 &\leq (32/31)^2(1/2 + 6/8^2)\Delta_t^2 + \frac{(32/31)^2 6\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{m^2} + (32/31)^2 6\alpha^2n^2\|\nabla f(\boldsymbol{w}_{t+1})\|_2^2 \\ &\leq (2/3)\Delta_t^2 + \frac{7\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{m^2} + 7\alpha^2n^2\|\nabla f(\boldsymbol{w}_{t+1})\|_2^2.\end{aligned} \tag{E.8}$$

We next sum (E.8) over $t \in [T-1]$ and add (E.7):

$$\begin{aligned}\Delta_1^2 + \sum_{t=2}^{T}\Delta_t^2 &\leq (2/3)\sum_{t=2}^{T}\Delta_{t-1}^2 + \frac{(T-1)7\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{m^2} + 3\alpha^2n^2\sigma^2 + 7\alpha^2n^2\sum_{t=1}^{T}\|\nabla f(\boldsymbol{w}_t)\|_2^2 \\ \frac{1}{T}\sum_{t=1}^{T}\Delta_t^2 &\leq (2/3)\frac{1}{T}\sum_{t=1}^{T}\Delta_t^2 + \frac{7\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{m^2} + \frac{3\alpha^2n^2\sigma^2}{T} + 7\alpha^2n^2\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\boldsymbol{w}_t)\|_2^2 \\ &\leq \frac{21\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{m^2} + \frac{9\alpha^2n^2\sigma^2}{T} + 21\alpha^2n^2\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\boldsymbol{w}_t)\|_2^2,\end{aligned}$$

yielding the claim. □

We next build on Lemma 8.

Lemma 9. *If $\alpha \leq \frac{2}{9n\mu}$, then, for $\rho = 1 - \frac{\alpha n \mu}{2}$,*

$$\sum_{t=1}^{T} \rho^{T-t} \Delta_t^2 \leq 12\rho^{T-1}\alpha^2 n^2 \sigma^2 + \frac{28\rho\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{(1-\rho)m^2} + \frac{1}{2L_{2,\infty}^2} \sum_{t=1}^{T} \rho^{T-t} \|\nabla f(\boldsymbol{w}_t)\|_2^2.$$

Proof. Recall (E.7) from Lemma 8:

$$\Delta_1^2 \leq 3\alpha^2 n^2 \sigma^2 + 3\alpha^2 n^2 \|\nabla f(\boldsymbol{w}_1)\|_2^2.$$

We multiply each term Δ_t with ρ^{T-t} for $t \in [T]$ and get

$$\rho^{T-1}\Delta_1^2 \leq \rho^{T-1} 3\alpha^2 n^2 \sigma^2 + \rho^{T-1} 3\alpha^2 n^2 \|\nabla f(\boldsymbol{w}_1)\|_2^2. \tag{E.9}$$

Similarly, recall (E.8) from Lemma 8,

$$\Delta_{t+1}^2 \leq (2/3)\Delta_t^2 + \frac{7\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{m^2} + 7\alpha^2 n^2 \|\nabla f(\boldsymbol{w}_{t+1})\|_2^2,$$

for which we also multiply each term Δ_t with ρ^{T-t} for $t \in [T]$, and get

$$\begin{aligned}\rho^{T-t}\Delta_t^2 &\leq (2/3)\rho^{T-t}\Delta_{t-1}^2 + \rho^{T-t}\frac{7\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{m^2} + \rho^{T-t}7\alpha^2 n^2 \|\nabla f(\boldsymbol{w}_t)\|_2^2 \\ &\leq (3/4)\rho^{T-(t-1)}\Delta_{t-1}^2 + \rho^{T-t}\frac{7\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{m^2} + \rho^{T-t}7\alpha^2 n^2 \|\nabla f(\boldsymbol{w}_t)\|_2^2, \quad \forall t \in \{2,\ldots,T\},\end{aligned} \tag{E.10}$$

where we have used $\alpha \leq \frac{2}{9n\mu}$ so that $\rho = 1 - \frac{\alpha n \mu}{2} \geq (2/3)(4/3)$.

Next, we sum the bounds in (E.9) and (E.10) for $\rho^{T-t}\Delta_t$ for all $t \in [T]$, and we get

$$\rho^{T-1}\Delta_1^2 + \sum_{t=2}^{T} \rho^{T-t}\Delta_t^2 \leq \frac{3}{4}\sum_{t=2}^{T} \rho^{T-(t-1)}\Delta_{t-1}^2 + \rho^{T-1} 3\alpha^2 n^2 \sigma^2 + \sum_{t=1}^{T} \rho^{T-t}\frac{7\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{m^2} + 7\alpha^2 n^2 \sum_{t=1}^{T} \rho^{T-t}\|\nabla f(\boldsymbol{w}_t)\|_2^2.$$

We can rewrite the right-hand side as

$$\leq \frac{3}{4}\sum_{t=1}^{T}\rho^{T-t}\Delta_t^2 + \rho^{T-1}3\alpha^2n^2\sigma^2 + \frac{7\rho\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{(1-\rho)m^2} + 7\alpha^2n^2\sum_{t=1}^{T}\rho^{T-t}\|\nabla f(\boldsymbol{w}_t)\|_2^2$$
$$\leq 12\rho^{T-1}\alpha^2n^2\sigma^2 + \frac{28\rho\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{(1-\rho)m^2} + 28\alpha^2n^2\sum_{t=1}^{T}\rho^{T-t}\|\nabla f(\boldsymbol{w}_t)\|_2^2.$$

Lastly, we use $\alpha \leq \frac{1}{\sqrt{56}nL_{2,\infty}}$ to get:

$$\sum_{t=1}^{T}\rho^{T-t}\Delta_t^2 \leq 12\rho^{T-1}\alpha^2n^2\sigma^2 + \frac{28\rho\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{(1-\rho)m^2} + \frac{1}{2L_{2,\infty}^2}\sum_{t=1}^{T}\rho^{T-t}\|\nabla f(\boldsymbol{w}_t)\|^2.$$

□

E.2.6 Proof of Theorems 6 and 7

Using the Lemmas above, we next prove our main results, presented in Section 6.4.

Proof of Theorem 6. The given learning rate α satisfies the constraints of Lemma 5 and Lemma 8.

Therefore,

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\boldsymbol{w}_t)\|_2^2 \leq \frac{2F_1}{\alpha nT} + L_{2,\infty}^2\left(\frac{21(\alpha(\varsigma+\sigma)\tilde{A})^2}{m^2} + \frac{9(\alpha n\sigma)^2}{T} + 21\alpha^2n^2\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\boldsymbol{w}_t)\|_2^2\right)$$
$$\leq \frac{4F_1}{\alpha nT} + \frac{42L_{2,\infty}^2(\alpha(\varsigma+\sigma)\tilde{A})^2}{m^2} + \frac{18L_{2,\infty}^2(\alpha n\sigma)^2}{T},$$

due to $\alpha \leq \frac{1}{\sqrt{42}nL_{2,\infty}}$.

We next derive the convergence rate. Let $\Gamma = \frac{42(L_{2,\infty}(\varsigma+\sigma)\tilde{A})^2}{m^2} + \frac{18L_{2,\infty}^2 n^2\sigma^2}{T}$. Then,

$$\frac{1}{T}\sum_{t=1}^{T} \|\nabla f(\boldsymbol{w}_t)\|_2^2 \leq \frac{4F_1}{\alpha nT} + \Gamma\alpha^2.$$

We then set $\alpha \leq \left(\frac{4F_1}{n\Gamma T}\right)^{1/3}$. So we will have $\alpha = \min\left\{\frac{1}{16L_{2,\infty}(2n+\tilde{A}/m)}, \left(\frac{4F_1}{n\Gamma T}\right)^{1/3}\right\}$ or

$$\frac{1}{\alpha} = \max\left\{16L_{2,\infty}(2n + \tilde{A}/m), \left(\frac{4F_1}{n\Gamma T}\right)^{-1/3}\right\}.$$

Substitute α:

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} \|\nabla f(\boldsymbol{w}_t)\|_2^2 &\leq \frac{4F_1}{nT}\left\{16L_{2,\infty}(2n + \tilde{A}/m) + \left(\frac{4F_1}{n\Gamma T}\right)^{-1/3}\right\} + \Gamma\left(\frac{4F_1}{n\Gamma T}\right)^{2/3} \\
&\leq \left(\frac{4F_1}{nT}\right)^{2/3}\Gamma^{1/3} + \frac{64F_1L_{2,\infty}(2 + \tilde{A}/(mn))}{T} \\
&\leq \left(\frac{4F_1}{nT}\right)^{2/3}\left(\frac{(\sqrt{42}L_{2,\infty}(\varsigma + \sigma)\tilde{A})^{2/3}}{m^{2/3}} + \frac{(\sqrt{18}L_{2,\infty}n\sigma)^{2/3}}{T^{1/3}}\right) \\
&\quad + \frac{64F_1L_{2,\infty}(2 + \tilde{A}/(mn))}{T} \\
&\leq \frac{(4\sqrt{42}F_1L_{2,\infty}(\varsigma + \sigma)\tilde{A})^{2/3}}{(mnT)^{2/3}} + \frac{(72F_1L_{2,\infty}\sigma)^{2/3}}{T} \\
&\quad + \frac{64F_1L_{2,\infty}(2 + \tilde{A}/(mn))}{T},
\end{aligned}$$

Since $(4\sqrt{42})^{2/3} < 9$, the above is

$$\leq \frac{9(F_1L_{2,\infty}(\varsigma + \sigma)\tilde{A})^{2/3}}{(mnT)^{2/3}} + \frac{(72F_1L_{2,\infty}\sigma)^{2/3} + 64F_1L_{2,\infty}(2 + \tilde{A}/(mn))}{T},$$

in which the leading term (slowest in terms of T) is $\tilde{O}((mnT)^{-2/3})$, proving the claim. □

Proof of Theorem 7. With the P.L. assumption (Assumption 5), we use Lemma 6 and Lemma 9. We show that their constraints are satisfied later) to get

$$\begin{aligned}
F_{T+1} &\leq \rho^T F_1 + \frac{\alpha n L_{2,\infty}^2}{2} \sum_{t=1}^{T} \rho^{T-t} \left(\Delta_t^2 - \frac{1}{2L_{2,\infty}^2} \|\nabla f(\boldsymbol{w}_t)\|_2^2 \right) \\
&\leq \rho^T F_1 + \frac{\alpha n L_{2,\infty}^2}{2} \left(12\rho^{T-1}\alpha^2 n^2 \sigma^2 + \frac{28\rho\alpha^2(\varsigma+\sigma)^2\tilde{A}^2}{(1-\rho)m^2} \right) \\
&\leq \rho^T F_1 + \rho^{T-1} 6\alpha^3 n^3 L_{2,\infty}^2 \sigma^2 + \frac{28\rho\alpha^3 n L_{2\infty}^2(\varsigma+\sigma)^2\tilde{A}^2}{\alpha n\mu m^2} \\
&\leq \rho^T F_1 + \rho^T 7\alpha^3 n^3 L_{2,\infty}^2 \sigma^2 + \frac{28\rho\alpha^3 n L_{2\infty}^2(\varsigma+\sigma)^2\tilde{A}^2}{\alpha n\mu m^2} \\
&\leq \rho^T (F_1 + \sigma^2/L_{2,\infty}) + \frac{28\alpha^2 L_{2,\infty}^2(\varsigma+\sigma)^2\tilde{A}^2}{\mu m^2} \\
&\leq (F_1 + \sigma^2/L_{2,\infty}) \exp(-T\alpha n\mu/2) + \frac{28\alpha^2 L_{2,\infty}^2(\varsigma+\sigma)^2\tilde{A}^2}{\mu m^2},
\end{aligned}$$

where we have further constrained $\alpha \leq \frac{2}{9n\mu}$ so that $\rho \leq 9/8$ in the forth inequality and $\alpha \leq \frac{1}{7^{1/3} n L_{2,\infty}}$ in the fifth inequality. By setting the derivative w.r.t α of the RHS to 0, the minimizer α under the constraint that $0 < \alpha \leq \min\left\{\frac{2}{9n\mu}, \frac{1}{16L_{2,\infty}(2n+\tilde{A}/m)}\right\}$ (required by the lemmas) is:

$$\alpha = \frac{2}{Tn\mu} W_0(T^2 m^2 n^2 C_3),$$

as long as

$$\begin{aligned}
T &\geq 1 + \frac{2}{n\mu} \max\{(9/2)n\mu, 16L_{2,\infty}(2n + \tilde{A}/m) W_0(T^2 m^2 n^2 C_3)\} \\
&= 10 + \frac{1}{\mu} 32 L_{2,\infty}(2 + \tilde{A}/(mn)) W_0(T^2 m^2 n^2 C_3),
\end{aligned}$$

where $C_3 = \frac{(F_1+\sigma^2/L_{2,\infty})\mu^2}{224 L_{2,\infty}^2(\varsigma+\sigma)^2\tilde{A}^2}$.

What we did here was to set T just large enough so that the minimizer α is the same with or without the constraint.

Denoting $\tilde{W} = W_0(T^2m^2n^2C_3) = \tilde{O}(1)$, we get

$$F_{T+1} \leq \frac{(F_1 + \sigma^2/L_{2,\infty})\tilde{W}}{T^2m^2n^2C_3} + \frac{112L_{2,\infty}^2(\varsigma+\sigma)^2\tilde{A}^2\tilde{W}^2}{T^2m^2n^2\mu^3}$$
$$\leq \frac{1}{T^2m^2n^2}\left(\frac{(F_1 + \sigma^2/L_{2,\infty})\tilde{W}}{\tilde{C}_3} + \frac{112L_{2,\infty}^2(\varsigma+\sigma)^2\tilde{A}^2\tilde{W}^2}{\mu^3}\right),$$

which shows rate the convergence rate in the P.L. case is $\tilde{O}((mnT)^{-2})$. □

E.3 Experiment Details

Here we provide more extensive details on our empirical results. This includes background information on our experimental setup in the main paper (Appendix E.3.1), an additional simulation experiment on pre-training and fine-tuning Tiny GPT-2 (Appendix E.3.2), and an additional simulation experiment that investigates CD-GraB with different learning rates (Appendix E.3.3). Our source code can be found here.

E.3.1 Additional details on setup for main paper experiments

Distributed experiments

We provide additional details on the experiments shown in Figure 6.3.

Hardware and software. We use a single machine with 128 GiB memory, 1 CPU, and 4 Nvidia GeForce 2080ti GPUs for the HMDA mortgage application, M4, and WikiText-2 tasks. We first discard the remainder $N \mod B$, and then

randomly partition n to each worker. Our experiments are all implemented with the PyTorch library.

Datasets and models.

- **Logistic regression on mortgage application (NY 2017 subset)**: The US Home Mortgage Disclose Act (HMDA) makes available US national data regarding mortgage applications, which has recently been packaged up for easy ML research use [141]. We use the binary classification version of the task, which classifies features as either "grant loan" or "deny loan," for the New York (NY) 2017 subset of the dataset, which includes 244107 examples with 18 features. We model this problem using logistic regression, for which we first perform a random 80/20 train/test split on the raw dataset, and then we discard $N \mod B$ (B is the aggregated minibatch size) examples to ensure that each worker receives exactly n examples. We use 1 worker per GPU, and in total we have $m = 4$ workers, and use NCCL [447] as the distributed communication backend; $m = 4,\ n = 48816,\ d = 18,\ B = 16$. We report test accuracy as our evaluation metric.
- **LSTM on WikiText-2**: We follow the settings in Lu et al. [384] and train a 2-layer LSTM with an embedding size of 32 and dropout set to 0. We use backpropagation through time, for which we set the sequence length to 35. We also adopt the word-vector-classifier-weight-sharing strategy inspired by Inan et al. [291]. WikiText-2 [562] has 600 articles in the train set, with more than 2M tokens and 30K vocabulary; the validation and test sets each have 60 articles. We adapt our training script from PyTorch's official Word Language Modeling Github repository. We use 4 workers in total, with each GPU hosting 1 worker, and use NCCL as the distributed

communication backend; $m = 4,\ n = 3728,\ d = 1081760,\ B = 16$. We report test perplexity as the evaluation metric, and we follow the HuggingFace's approach of computing perplexity as the exponentiated average negative log-likelihood of a sequence.[1]

- **Autoregressive MLP on M4 Weekly Dataset**: We build a 3-layer autoregressive MLP with a hidden dimension of 64. We set input sequence length to be 20 and the output sequence length to be 6. M4 is a time series dataset composed of 100,000 time series for yearly, quarterly, monthly, weekly, daily and hourly data [391], which is drawn from a random sample of ForeDeCk database [556]. We use the weekly data in our experiment. We use 32 workers, where each of the 4 GPUs hosts 8 process workers. We use GLOO as the distributed communication backend. $m = 32,\ n = 3355,\ d = 5569,\ B = 32$. We report test symmetric mean absolute percentage error (SMAPE) as the evaluation metric. We follow the formula of SMAPE in [391] as follows:

$$\text{SMAPE} \triangleq \frac{2}{h} \sum_{t=n+1}^{n+h} \frac{|Y_t - \hat{Y}_t|}{|Y_t| + |\hat{Y}_t|} * 100\%$$

  where Y_t is the reference time series value at timestep t, $\hat{Y}_t$ is the forecast time series value at timestep t, and h is the forecasting horizon and n is the number of datapoints.

Hyperparameter optimization. For all tasks, we tune the learning rate α for D-RR first, and then use the selected learning rate for CD-GraB. Therefore, a performance improvement here implies we would have in-place substitution benefits via switching from D-RR to CD-GraB with identical learning rate and

[1] `https://huggingface.co/docs/transformers/perplexity`

experiment setups. We use SGD with momentum as the optimizer for all tasks. The hyperparameters for each task are as follows:

- **Logistic regression on mortgage application (NY 2017 subset)**: α = 5e-3 $\in$ {1e-2, 5e-3, 1e-3}, momentum: 0.9, weight decay: 0, B: 16.
- **LSTM on WikiText-2**: $\alpha = 5 \in \{5, 10\}$ and decays by 0.1 per 10 epochs, momentum: 0.9, weight decay: 0, B: 16.
- **Autoregressive MLP on Weekly M4 Dataset** α = 1e-3 $\in$ {1e-2, 1e-3, 1e-4}, momentum: 0.9, weight decay: 0, B: 32.

Memory Overhead of CD-GraB in LSTM on WikiText-2 Task

We profile the CUDA memory usage for the LSTM on WikiText-2 Task with CD-GraB and D-RR to understand the memory overhead of both data permutation algorithms. This memory analysis is both task and implementation dependent, but still serves to illustrate the overarching point that CD-GraB's memory overhead is not so significant. The additional overhead comes from two sources for CD-GraB: communication and example sorting (Figure E.4).

In more detail: in our LSTM experiment, each local worker will share its gradients with all other workers at every optimization step. To reduce the communication burden of CD-GraB, we make each local worker function as an order server.[2] The memory consumption of forward, backward, and optimizer states between CD-GraB and D-RR should be (at least approximately) identical. The

[2]An ideal location for a dedicated order server is on a network node that has large input bandwidth and memory buffer to host all gradients while not blocking the normal optimization stages. Since we do not have enough computational resources to host a dedicated order server, we make each worker an order server.

model size of LSTM is roughly 4 MiB. We use 4 workers, and as each worker (functioning as an order server) needs to *all-gather* gradients, the memory overhead for *all-gather* communication is roughly tensor_size $\times$ # workers = 4 MiB $\times$ 4 = 16 MiB for CD-GraB(we observe 16.51 MiB in practice, Communication in Figure E.4), while D-RR only needs to *all-reduce* the gradients (yielding no memory overhead; the communication buffer for *all-reduce* is reusing the same gradient tensor). The PairBalance algorithm (Algorithm 6) internally needs a model-sized accumulator as the running sum $\boldsymbol{r}$, and both computing inner product between $\boldsymbol{r}$ and $\boldsymbol{g}_1 - \boldsymbol{g}_2$ and updating $\boldsymbol{r}$ with $\boldsymbol{r} + s\boldsymbol{c}$ takes virtually no space with a memory-efficient implementation. Therefore, the memory consumption for PairBalance is still roughly 4 MiB (Data Sorter in Figure E.4).

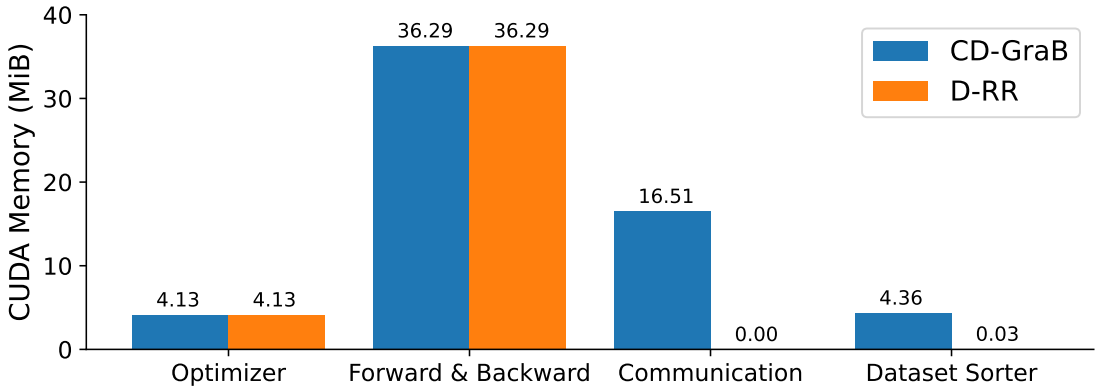


Figure E.4: CUDA Memory Overhead of CD-GraB and D-RR in LSTM on WikiText-2 Task.

The main memory overhead of CD-GraB will be dominated by the communication buffer size on the order server side: the order server have to gather the gradient (differences) from all workers, and sequentially apply the PairBalance algorithm. This memory bottleneck would similarly be found in GraB as GraB also needs per-example gradients to perform Balance sequentially.

A future algorithmic improvement to the general gradient balancing frame-

work would be finding a balancing algorithm that does not need per-example gradients to achieve comparable convergence guarantees. However, we still notice that PairBalance is more memory-efficient than Balance as Balance needs to store 3 model-sized tensors: 1 for the balancing accumulator, 1 for running-average gradients for last epoch, and 1 for the running-average for current epoch. In contrast, PairBalance only needs 1 model-sized tensor as the balancing accumulator.

Simulated ablation study using LeNet on CIFAR-10

In the experiment shown on Figure 6.4, we select the same learning rate, momentum, and weight decay as the LeNet experiment in Lu et al. [384]. We use 3 different random seeds to control 3 different initialization and the randomness in random reshuffling. The aggregated minibatch size B is 64 for all runs. We implement this ablation study by using 1 GPU with up to $m = 64$ workers (processes). As above, we discard $N \mod B$ examples and partition the remaining examples evenly on each worker.

α = 1e-3 $\in$ {1e-2, 5e-3, 1e-3, 5e-4, 1e-4}, momentum: 0.9, weight decay: 1e-2, B: 64.

We do not implement this via distributed environment due to the fact that we do not have access to 64 GPUs, but expect the simulation results to be a good reflection of the results we would obtain in a multi-GPU setting.

Parallel herding bound. We further investigate the empirical parallel herding bounds (6.8) for the LeNet experiment for the different ordering methods. We plot the results in Figure E.5. We observe that as the number of workers

Figure E.5: Empirical parallel herding bounds of gradients for each algorithm in LeNet experiment. We plot the mean as the curve and standard deviation across 3 random seeds.

increases, the empirical parallel herding bounds of both **ID-GraB (Bal)** and **ID-GraB (PairBal)** also increase, and eventually exhibit little difference with D-RR. CD-GraB, in contrast, exhibits a consistently lower bound.

For comparison, we also run a simulation experiment on synthetic data to investigate the behavior of the parallel herding bound. We include these below, in Figure E.6.

We randomly initialize 1 million random vectors $z_{i,j}$ from a uniform distribution between 0 and 1 with 16 dimensions as $z_{i,j} \sim \mathsf{Unif}(0,1)^{16}$, and then we zero-center this set of 1 million vectors and normalize them to all have L_2 norm as 1. We then evenly partition this set of 1 million random vectors to $\{5, 10, 20, 50, 100\}$ workers and run each example ordering algorithm.

In Figure E.6, we run CD-GraB, D-RR, **ID-GraB (Bal)**, **ID-GraB (PairBal)** on these random vectors, and compute the parallel herding bounds (6.8). From left to right in Figure E.6, we observe that as the number of workers m increases, the

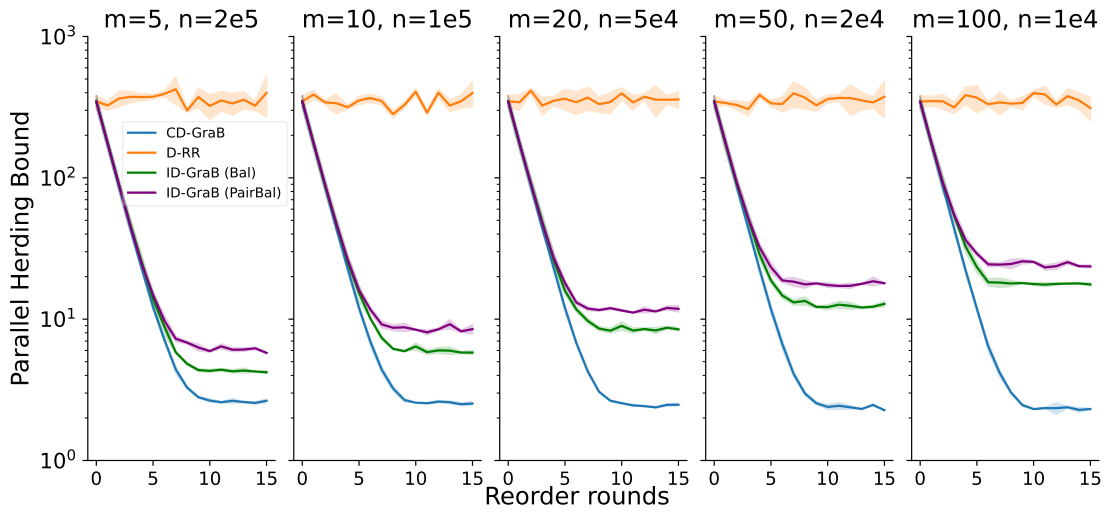


Figure E.6: Parallel herding bounds for different example ordering algorithms on N=1 million random vectors. We use 3 random seeds, plot the mean and standard deviation across each random seed as the shaded area.

parallel herding bound of **ID-GraB (Bal)**, **ID-GraB (PairBal)** becomes larger. This shows the importance of coordination when we have a large number of workers.

These results for random vectors cohere with our above results for LeNet on CIFAR-10.

E.3.2 An additional simulation experiment: pre-training and fine-tuning Tiny GPT-2

We perform an end-to-end simulation experiment involving pre-training and fine-tuning Tiny GPT-2 on WikiText-103, which we document below.

Pre-training

We adapt the training script from the HuggingFace's PyTorch casual language modeling code to train the GPT-2 architecture [483]. We set the maximum sequence length to 128 and token and positional embedding dimension to 128; use 2 hidden layers in the transformer encoder and 2 attention heads; and disable dropout. This model configuration corresponds to the following Python code snippet:

```
from transformers import GPT2Config, GPT2LMHeadModel, GPT2Tokenizer

tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
config = GPT2Config.from_pretrained('gpt2')
config.n_embd = 128
config.n_ctx = 128
config.n_layer = 2
config.n_head = 2
config.n_positions = 128
config.summary_first_dropout = 0
config.attn_pdrop = 0
config.resid_pdrop = 0
model = GPT2LMHeadModel(config)
```

We train our Tiny GPT-2 model from scratch on WikiText-103 [562]. WikiText-103 is a standard language modeling benchmark that has 28,475 articles in the train set, and 60 for both the validation and test sets, with more than 100M tokens and 267K vocabulary inside the train set. We use the original GPT-2 tokenizer, and use maximum sequence length 128. We note that this is much smaller than the default maximum sequence length for GPT-2, which is 1024, which was too large to use given our computational budget. Nevertheless, 128

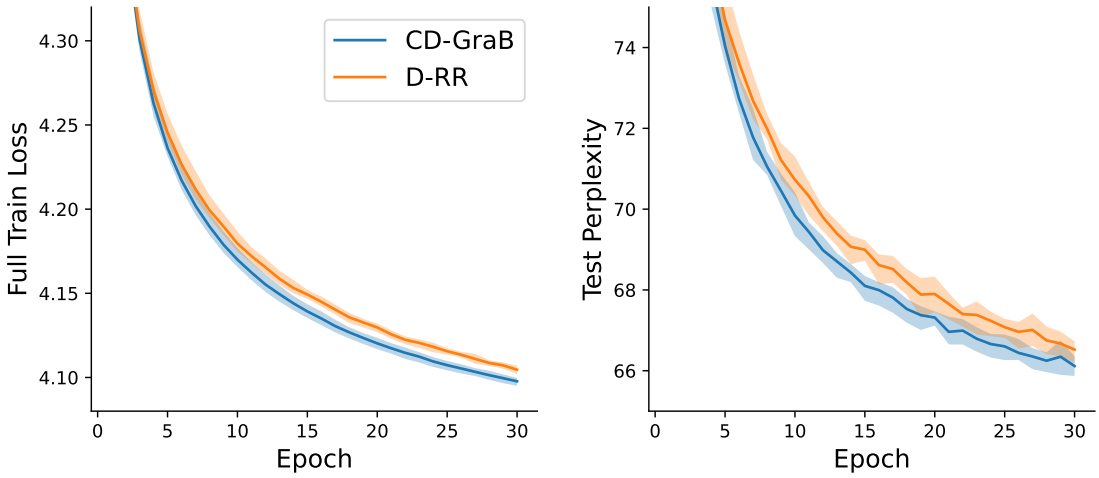


Figure E.7: Pre-training Tiny GPT-2 on WikiText-103 from scratch: Convergence for CD-GraB and D-RR with $m = 64$ workers. The aggregated minibatch size per update is 64. We use 3 random seeds, and plot the mean and standard deviation.

is still a reasonable sequence length for the initial phrase of pre-training; BERT uses a sequence length of 128 for the first 90% of pre-training steps to speedup the experiment [171]. We tune the learning rate for D-RR with the grid { 5e-3, 1e-3, 5e-4, 1e-4 } (the final learning rate is 5e-4), and use AdamW optimizer [380]. We use 3 random seeds. Before the training, we simulate 64 workers, and similarly divide the training dataset evenly across them by discarding $N \mod B$ examples. Our hyperparameter optimization space is listed below:

Pretraining Hyperparameters. α =5e-4 $\in$ {5e-3, 1e-3, 5e-4, 1e-4}, weight decay: 1e-4, B: 64.

We document convergence for pre-training in Figure E.7, and use test perplexity as our evaluation metric.

Fine-tuning

We then fine-tune the pre-trained Tiny GPT-2 model on downstream tasks. For each task, we load the pre-trained foundation model weights obtained at the end of 30 epochs of each example ordering algorithm after pretraining, and use the same example ordering algorithm to perform supervised fine-tuning. We focus on the largest 4 GLUE tasks [611]: MNLI, QQP, QNLI, and SST2. We tune the learning rate for D-RR with the AdamW optimizer, and for each run we report the best validation accuracy. We then take an average results of each run and summarize them in Table E.1. Our training script is adapted from the HuggingFace's PyTorch GLUE fine-tuning example codes.

Fine-Tuning Hyperparameters

- **MNLI** α =5e-4$\in$ {5e-3, 1e-3, 5e-4, 1e-4}, Weight decay: 1e-4, B: 32, epochs: 10, linear learning rate scheduler
- **QQP** α =5e-4$\in$ {5e-3, 1e-3, 5e-4, 1e-4}, Weight decay: 1e-4, B: 32, epochs: 10, linear learning rate scheduler
- **QNLI** α =5e-4$\in$ {5e-3, 1e-3, 5e-4, 1e-4}, Weight decay: 1e-4, B: 32, epochs: 10, linear learning rate scheduler
- **SST2** α =5e-4$\in$ {5e-3, 1e-3, 5e-4, 1e-4}, Weight decay: 1e-4, B: 32, epochs: 10, linear learning rate scheduler

We include these fine-tuning results in part to support our claim in Section 6.6 that CD-GraB exhibits its benefits more clearly when there are more training epochs. Our pre-training results suggest that CD-GraB would confer benefits to pre-training large models over multiple epochs; however, CD-GraB

	MNLI (Matched)	MNLI (Mismatched)	QQP	QNLI	SST2
CD-GraB	65.91 ± 0.46 %	64.36 ± 2.03 %	82.25 ± 0.21 %	62.11 ± 0.70 %	82.65 ± 0.39 %
D-RR	65.42 ± 0.36 %	63.93 ± 1.63 %	81.74 ± 0.33 %	61.87 ± 0.67 %	82.68 ± 0.57 %

Table E.1: GLUE fine-tuning datasets: Validation accuracy of CD-GraB in comparison to D-RR, reporting mean and standard deviation of best results for each run. There are 3 runs for each example ordering algorithm.

will not necessarily be useful for short runs of fine-tuning (as indicated in Table E.1, for which the results for both ordering algorithms are effectively identical).

E.3.3 Ablation simulation study: The impact of learning rate α

In the experiment shown on Figure E.8, we select the same momentum and weight decay as the LeNet experiment for 3 random seeds as in Appendix E.3.1. The aggregated minibatch size is still 64 for all runs, and we use 64 workers.

We find that when we increase the learning rate from 1e-3 to 1e-2, CD-GraB still maintains relatively better performance than D-RR. The best learning rate for D-RR is 1e-3, in terms of achieving the best test accuracy. We did not tune the learning rate for CD-GraB, and we expect that it is possible to use a higher learning rate and still maintain better empirical performance than D-RR and even faster convergence. We defer such empirical investigations to future work. Altogether, these preliminary empirical results confirm that it is possible to use higher learning rate for CD-GraB, given that online PairBalance does not need to use a stale mean (Section 6.3.2), which would make larger learning rates perform poorly.

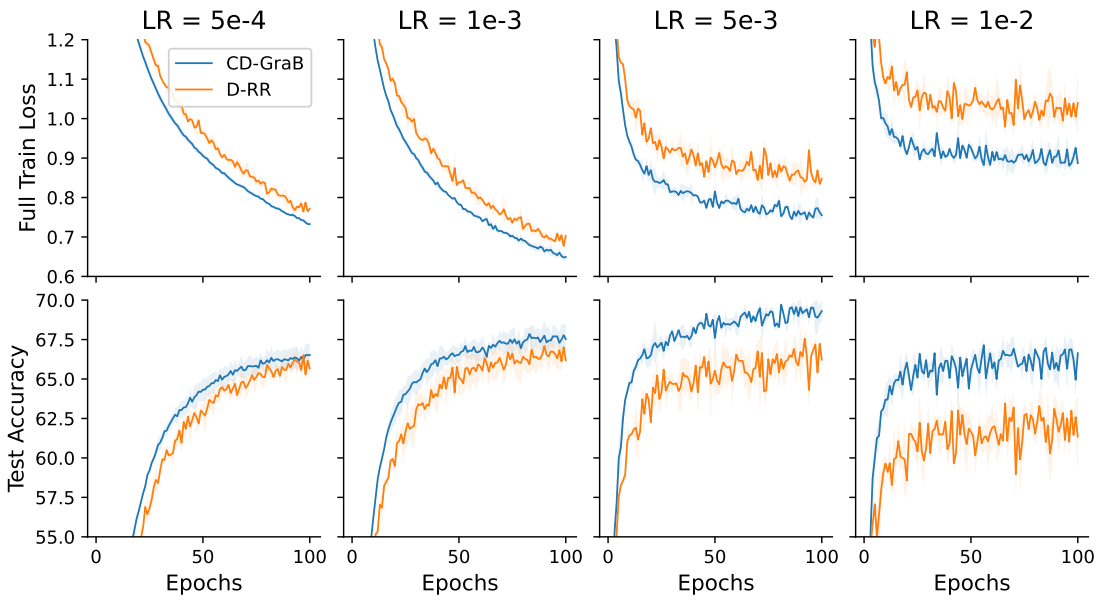


Figure E.8: Convergence for CD-GraB, D-RR training LeNet on CIFAR-10, with $m = 64$ workers. The aggregated minibatch size per update is 64. We use 3 random seeds, and plot the mean values across random seeds as the curve, the standard deviation as the shaded area.

APPENDIX F

APPENDIX FOR COMMONCANVAS

F.1 Details on Data Scarcity Analysis

F.1.1 Hypothesis: Diffusion models are too small

A back-of-the-envelope calculation provides some insight on why this is the case. Consider a training dataset consisting of N images with resolution $H \times W$ and c channels. To completely memorize the training data, the model must be capable of storing $c \times H \times W \times N$ numbers. Given a number of trainable parameters N_p, it is natural to assume that on average each parameter is capable of storing roughly enough information to reconstruct a single number from the training dataset. Under this assumption, complete memorization is only possible if the size of the training dataset is at or below a critical size N_c ($N \leq N_c$) with N_c given by $N_c = \frac{N_p}{cHW}$. Note that this critical size assumes the data cannot be further compressed, which is obviously not the case for natural images. However, SD2 and SDXL are latent diffusion models, which first use a pretrained encoder to compress images by a factor of 8 in both H and W, and so when we train LDMS like SD2 and SDXL, we are training on data that has been significantly compressed already.

In our experiments, $c = 4$ and $H = W = 32$, corresponding to 256×256 resolution RGB images in the SD2 and SDXL latent space. The SD2 UNet has $N_p = 866 \times 10^6$ trainable parameters, and SDXL's UNet has $N_p = 2567 \times 10^6$. So we calculate $N_c \approx 0.2 \times 10^6$ for SD2 and $N_c \approx 0.6 \times 10^6$ for CommonCanvas-Large;

both of these numbers are several orders of magnitude below the size of our YFCC derived datasets, and so even with significant additional data compression we expect that our CommonCatalog datasets should be sufficient to train both SD2 and SDXL. Additionally, this argument predicts that we should only begin to see significant overfitting in these models for datasets of size $N \sim 10^6$. These estimates are resolution dependent, and as image resolution increases we expect that N_c will decrease as more information is provided per image.

F.1.2 Increasing model capacity

We also train a variant of SD2 with more trainable parameters, taking the UNet from SDXL. We refer to this model as CommonCanvas-LNC. We adapt the SDXL UNet architecture to SD2 by changing the cross-attention dimensionality to match that of the SD2 text encoder hidden state dimensionality (1024 for SD2 vs. 2048 for SDXL). SDXL also retrains the VAE component in their model, and we use this improved performance VAE as well. Except for these changes, the architecture is identical to that of SD2.

F.2 Training Dataset and Model Details

F.2.1 LAION-2B

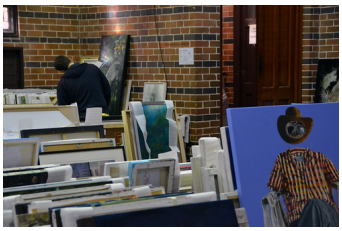
The fact that LAION is not a stable benchmark can lead to multiple reproducability and security issues. Data poisoning attacks would be difficult to detect at the scale of 2 billion parameters. While this could be mitigated by using hash

Table F.1: CC licenses in YFCC100M.

CC License	# Images	% Captioned
CC-BY-NC-ND-2.0	25,790,117	33.52%
CC-BY-ND-2.0	4,827,970	30.23%
CC-BY-NC-2.0	12,468,229	31.39%
CC-BY-NC-SA-2.0	28,314,685	31.57%
CC-BY-SA 2.0	9,270,079	34.05%
CC-BY 2.0	16,962,338	28.96%

Table F.2: Randomly sampled images from the YFCC [579] training set. Our synthetic BLIP2 captions are also provided below.

a person riding a bike on a dirt road



a paintings on the wall

an orange and blue race car driving on a track

values of the images, then any time the a site decide to re-encode the image, those images would now need to be excluded from the dataset. Furthermore, targeted data poisoning attacks for diffusion models are no longer just academic conjecture. Last year after the release of Stable Diffusion, a protest was launched on ArtStation that had uses upload images that said "NoAI" to taint future training data for generative models after artists felt as though their work had been unfairly used to train the models. With the high degree of link rot, targeted attacks are fairly easy. Furthermore, reproduction of the experiments becomes virtually impossible. This means any benchmarks that use copies of LAION as ground truth are are likely using differing subsets of the full dataset.

Table F.3: Top 10 highest frequency captions in the YFCC dataset. The most common captions are not user generated and are not very descriptive of the corresponding image.

YFCC Original Caption	Count
OLYMPUS+DIGITAL+CAMERA	184889
SONY+DSC	123128
Exif_JPEG_PICTURE	104480
Barclays+Center+Arena%0AAtlantic+Yards%0A6th+and+Atlantic+A	68832
Olympus+digital+camera	54805
Effortlessly+uploaded+by Eye-Fi	48388
.	43227
-+Camera+phone+upload+powered+by ShoZu	38856
Sony+dsc	32709
Photo+by @Kmeron —Facebook page is this way—	23754

Sourcing Creative-Commons images

We source these images from YFCC100M. ND means derivative works are not licensed or the license doesn't allow the user to create derivative works. NC means images cannot be used in commercial contexts. CommonCatalog-C only contains data from the bottom two (yellow) rows, reflecting images licensed for commercial contexts (i.e., roughly 25 million images). CommonCatalog-NC contains CommonCatalog-C, and additionally includes the middle two (blue) rows, reflecting images licensed for non-commercial purposes. We do not include the roughly 30 million images in the top two (pink) rows in CommonCatalog, as they are non-derivative licenses. We do not train on these images. We do, however, produce BLIP-2 captions for them and release those captions as an evaluation set.

Table F.4: Number of usable captions from OpenAI's YFCC14M dataset [480]. This table is actually a subset from F.1 for which either the user description or image title were deemed usable. These figures provide an estimate on how many images in each category are actually potentially usable as captions.

License Name	count
CC-BY 2.0	2448002
CC-BY-ND 2.0	682273
CC-BY-NC 2.0	1925854
CC-BY-NC-ND 2.0	4058817
CC-BY-NC-SA 2.0	4146113
CC-BY-SA 2.0	1568336

F.2.2 Model Architecture

We follow the model architecture and training recipe of Stable Diffusion 2 as closely as we can to best reproduce the model for CC-Small. The model has an identical number of params and structure as the original model. In fact, we can even load SD2's model weights into our framework due to the identical architecture and naming scheme. We are able to achieve virtually identical performance with SD2 in a much shorter training time with less data. We use the same VAE, tokenizers, and UNet archicture as SD2 except for reducing the precision of the normalization layers.

Our CC-Large model takes SD2's model and replaces the UNet with the SDXL architecture [473]. Like CC-Small, we also replace the normalization layers with their low-precision version. The replacement of all the normalization layers is handled automatically by MosaicML's Composer library [422]. We perform all dataloading through MosaicML's streaming library [423].

F.2.3 Release and documentation

We release CommonCatalog and information about how to download CommonCanvas at `https://github.com/mosaicml/diffusion/blob/main/assets/common-canvas.md`, with an associated data sheet.

F.3 Telephoning

We dub our solution for handling the lack of captions in CC images as *telephoning*, a type of transfer learning (Figure 9.3). Telephoning assumes the existence of a large labeled dataset $\mathcal{D}_1 = \{(x^{(i)}, y^{(i)})\}_{i=1}^n$, consisting of pairs of high-dimensional $x^{(i)}$ (e.g., images, audio) that map to a compact, structured label $y^{(i)}$ (e.g., caption, audio transcript). Telephoning trains a forward model $q(y|x)$ on $\mathcal{D}_1$ to learn the mapping of y given x via maximum likelihood learning $\max_{q \in Q} \sum_{i=1}^n \log q(y^{(i)}|x^{(i)})$. It then uses q as training signal for a reverse model $p(x|y)$ trained on a *separate* dataset $\mathcal{D}_2 = \{x^{(i)}\}_{i=1}^m$ by maximizing $\sum_{i=1}^m \mathbb{E}_{y \sim q(y|x^{(i)})}[\log p(x^{(i)}|y^{(i)})]$, the likelihood of the data $\mathcal{D}_2$ and the predicted label y under q. This forms a type of knowledge transfer from the forward labeling task defined by $\mathcal{D}_1$ to the reverse task of inverting x from y on a separate $\mathcal{D}_2$.

While telephoning can be viewed as a type of synthetic labeling, it becomes particularly interesting when x is a type of protected modality (e.g., a copyrighted image), while y is a compact representation of x that does not encode sensitive aspects of y (e.g., a generic caption). Effectively, telephoning performs a type of "lossy compression" or "distillation" from a high-dimensional or information-rich x (e.g., an image of Snoopy) to a low-dimensional or

information-poor y that loses the sensitive content in x (e.g., the visual characteristics of Snoopy). Because this compression step is "lossy", a reconstruction x' of x from $p(x|y)$ via y often does not remotely resemble the original input, just like in a game of telephone [393]. We derive the term telephoning from the above intuition, and employ it as useful shorthand to denote instances of transfer learning that solve data-scarcity problems in multimodal generative modeling.

Telephoning for text-to-image modeling. In this work, we apply telephoning to the image and text domains, where CC images are the high-dimensional inputs x, and we use a pre-trained BLIP-2 model [369] for "lossy compression" to short-text captions y (Figure 9.3a). Together, these CC-image-caption pairs comprise the CommonCatalog dataset, which we use to train our CommonCanvas T2I models (Figure 9.3b). Even though BLIP-2 was pre-trained on LAION-400M [522], CommonCatalog and CommonCanvas never have direct access to LAION-400M or, importantly, anything that is similar to the images that BLIP-2 was trained on. Instead, we only have access to the mapping in the model, which, given an image input, produces lossy output text that inherently does not literally resemble its image counterpart (Figure 9.3c).

We draw on the example of Snoopy from [510]. Figure 9.3's Snoopy is CC-licensed [524].

Table F.5: Performance (throughput) and approximate cost of training SD2 UNet with our optimizations. Depending on the number of GPUs used, the cost to train the same models without these optimizations range from $90,000-$140,000

Number of A100s	256x256 (img/s)	512x512 (img/s)	512x512 with EMA (img/s)	Days to Train	Cost ($)
8	1100	290	290	101.04	$38,800.00
16	2180	585	580	50.29	$38,630.00
32	4080	1195	1160	25.01	$38,420.00
64	8530	2340	2220	12.63	$38,800.00
128	11600	4590	3927	6.79	$41,710.00

F.4 Details on Efficiency Optimizations

In this section we provide additional details on the optimizations we implemented to achieve SD2 training speedups. We also report the approximate cost of training our implementation of SD2 on various hardware configurations in Table F.5.

Flash Attention. Cross attention operations are a very expensive part of training that occurs in dozens of layers in diffusion model UNets [499]. Flash Attention is an efficient implementation that is optimized to work well with reduced precision and GPU hardware [163], which was implemented using the XFormers library [353], allowing us to save compute and memory usage.

Precomputing latents. Each forward pass of SD2 requires computing a latent representation of the input image, as well as transforming the caption into a text embedding. Instead of computing the latents for each example during training, we can precompute latents for the entire dataset, amortizing the cost. Doing so speeds up training of the model, especially at lower resolutions, in exchange for a one-time fixed cost of precomputing all the latents over 1 epoch.

Reduced-precision GroupNorm and LayerNorm. Most layers in SD2 are implemented in `float16` precision, but GroupNorm and LayerNorm are imple-

mented in float32, in part because it was assumed to be necessary for training stability. The resulting, frequent upcasting causes a major bottleneck in training speed. Recent work shows that it is safe to implement LayerNorm using float16 precision [474], and we found the same to be true of GroupNorm. We thus cast all GroupNorm and LayerNorm operators to float16 and are able to further reduce total memory consumption and accelerate training.

Fully-Sharded Data Parallelism (FSDP). FSDP is a variant of data-parallel training that shards the models parameters, gradients and optimizer state across multiple devices. When training data batches do not fit into memory, we do several forward and backward passes on smaller microbatches, followed by a single gradient update. At GPU scale, there may only be a single microbatch, so the time for the gradient update can become a significant bottleneck. In standard data distributed training, each GPU communicates all its gradients to every other GPU, and then each GPU updates its local copy of the model. Instead, we use a different paradigm inspired by [628] where each GPU only gets the gradients and updates the weights for a small part of the model before sending the updated weights for that part of the model to all of the other GPUs. By dividing the update step across all the GPUs, we can ensure that the amount of work per GPU decreases as we increase the number of GPUs, helping us achieve linear scaling. To tackle this problem, we use PyTorch's experimental support for Fully Sharded Data Parallelism (FSDP), specifically, FSDP's SHARD_GRAD_OP mode.

Scheduled Exponential Moving Average (EMA). SD2 uses EMA, which maintains an exponential moving average of the weights at every gradient update for the entire training period. This can be slow due to the memory operations

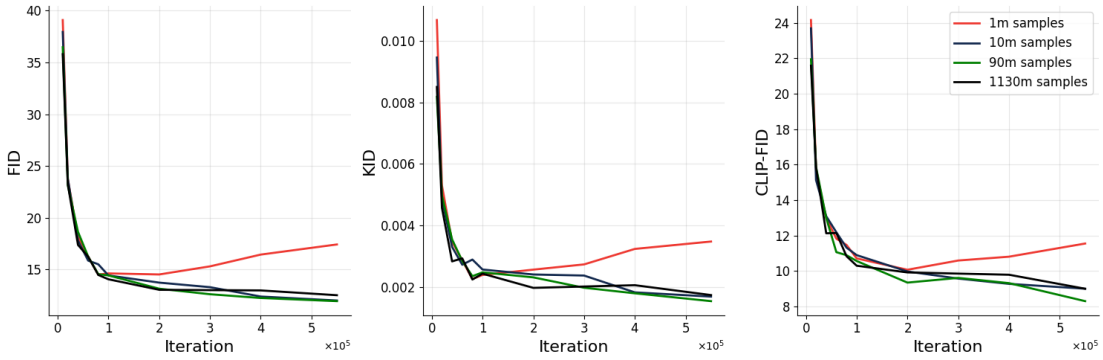


Figure F.1: MS COCO metrics over training duration for various dataset sizes. We investigate how reducing the size of the training dataset affects training dynamics, and find that performance is largely unchanged until dropping below 10 million samples. We show that the FID of the eval set remains stable as training progresses. However, reducing the number of samples in our training dataset to 1 million leads to divergence. This finding suggests that only 10 million to 1 million synthetic image caption pairs are needed for good performance on MS COCO.

required to read and write all the weights at every step. Since the old weights are decayed by a factor of 0.9999 at every batch, the early iterations of training only contribute minimally to the final average. We decide to only apply EMA for the final 50K steps (about 3.5% of the training period), and are able to avoid adding overhead and still achieve a nearly equivalent EMA model.

F.5 Additional Figures

We provide some additional details on training and qualitative results.

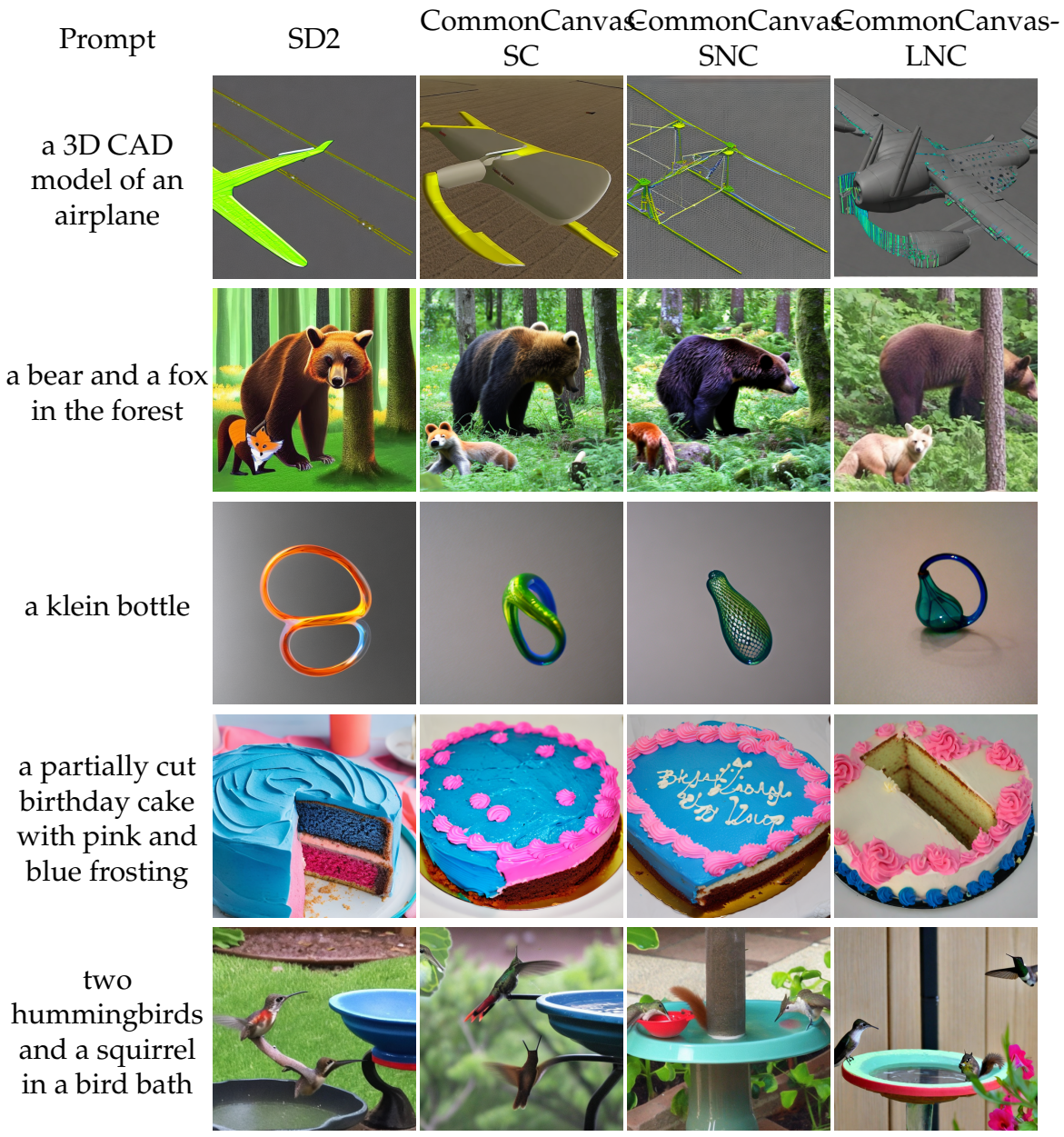


Figure F.2: Additional qualitative examples comparing SD2 to our model trained on the commerical split (CommonCanvas-SC), non-commerical split (CommonCanvas-SNC), and the larger UNet model trained on the non-commercial (CommonCanvas-LNC).

Figure F.3: Additional qualitative examples of our CommonCanvas models.

Figure F.4: Additional qualitative examples comparing our CommonCanvas models to SD2, given synthetic BLIP2 captions as prompts. While not perfect, our models are better at avoiding generating potentially problematic data.

APPENDIX G

ACCOUNTABILITY IN AN ALGORITHMIC SOCIETY: RELATIONALITY, RESPONSIBILITY, AND ROBUSTNESS IN MACHINE LEARNING

We begin with some additional framing for the rest of this dissertation. The work in this chapter represents ideas around accountability and machine-learning based systems that started taking shape in late 2017/ early 2018.[1] The contents of this chapter reflect the synthesis of these ideas in early 2022. Of course, the world changed significantly just months later with the public launch of ChatGPT. Nevertheless, the core points articulated in this piece hold up, and in a sense presaged, contemporary challenges of accountability and generative-AI systems. Arguably, this piece is even more salient now than it was at the time of writing. We will address this in future work.

Chapter summary: In 1996, *Accountability in a Computerized Society* [445] issued a clarion call concerning the erosion of accountability in society due to the ubiquitous delegation of consequential functions to computerized systems. Nissenbaum [445] described four barriers to accountability that computerization presented: 1) *many hands*, the problem of attributing moral responsibility for outcomes caused by many moral actors; 2) *"bugs"*, the way software developers might shrug off responsibility by suggesting software errors are unavoidable; 3) *computer as scapegoat*, the shifting of blame to computer systems as if they were moral actors; and 4) *ownership without liability*, the free pass given to the tech industry to deny responsibility for the software they produce We revisit these barriers in relation to the ascendance of data-driven algorithmic systems — i.e., machine learning (ML) or artificial intelligence (AI) — to uncover new challenges

[1] Indeed, the genesis of these ideas grounded my decision to go to graduate school in computer science, and to study topics at the intersection of machine learning and law.

for accountability that these systems present. Nissenbaum's original paper grounded discussion of the barriers in moral philosophy; we bring this analysis together with recent scholarship on relational accountability frameworks and discuss how the barriers present difficulties for instantiating a unified moral, relational framework in practice for data-driven algorithmic systems. We conclude by discussing ways of weakening the barriers in order to do so.

This chapter is a licensed derivative copy of work published at *FAccT 2022* [146].

G.1 Introduction

In 1996, writing against the backdrop of the meteoric rise of the commercial Internet [356], Nissenbaum [445] warned of the erosion of accountability due to four barriers inimical to societies increasingly reliant on computerized systems. These barriers are: *many hands*, to refer to the problem of attributing moral responsibility for outcomes caused by multiple moral actors; "*bugs*," the way software developers might shrug off responsibility by suggesting software errors are unavoidable; *computer as scapegoat*, the shifting of blame to computers as if they were moral actors; and *ownership without liability*, the free pass to the software industry to deny responsibility, particularly via shrink-wrap and click-wrap Terms of Service agreements. Today, twenty-five years later, significant work has been done to address the four barriers through developments in professional practices of computer science [297, 593], organizational management [299], and civil law [191, 425]; however, the effort to restore accountability remains incomplete. In the interim, the nature of computerized systems has been radically transformed by the ascendance of data-driven algorithmic sys-

tems[2] — e.g., machine learning (ML) and artificial intelligence (AI) — which either have replaced or complemented rule-based software systems, or have been incorporated within them as essential elements [49, 102, 334, 426, 640].

The resurgent interest in accountability is therefore timely for a world in which data-driven algorithmic systems are ubiquitous.[3] In domains as varied as finance, criminal justice, medicine, advertising, entertainment, hiring, manufacturing, and agriculture, these systems are simultaneously treated as revolutionary, adopted in high-stakes decision software and machines [16, 22, 26, 317, 420], and as novelties [316]. The failure to comprehensively establish accountability within computational systems through the 1990s and 2000s has thus left contemporary societies just as vulnerable to the dissipation of accountability, with even more at stake. We remain in need of conceptual, technical, and institutional mechanisms to assess how to achieve accountability for the harmful consequences of data-driven algorithmic systems — mechanisms that address both *whom* to hold accountable and *how* to hold them accountable for the legally cognizable harms of injury, property loss, and workplace hazards, and the not-yet-legally-cognizable harms increasingly associated with these systems, such as privacy violations [130], manipulative practices [12, 332], and automation-driven discrimination [16].

In light of growing concerns over accountability in computing, our paper revisits Nissenbaum's "four barriers to accountability" to assess whether insights from that work remain relevant to data-driven algorithmic systems, and to consider how the ascendance of such systems complicates, challenges, and demands more of sociotechnical, philosophical, and regulatory work. We first

[2]Since rule-based software systems are also "algorithmic," we specify which of the meanings we intend in settings where the context does not disambiguate.

[3]We adopt the term "Algorithmic Society" as used in Balkin [35].

provide context on recent developments in standards of care, law and policy, and computer science that are necessary for our analysis (Section G.1.1). Equipped with this background, we recapitulate the elements of moral philosophy on which Nissenbaum [445] depended (Section G.2.1), and discuss how this moral conception of accountability can be unified with Bovens's relational definition of accountability in political theory [81], which has drawn recent attention in AI ethics scholarship. In particular, we contend that moral and relational accountability can be brought together to illuminate the necessary parameters of an accountability framework for data-driven algorithmic systems — determining *who* is accountable, *for what*, *to whom*, and *under which circumstances* (Section G.2.2). To instantiate such a framework, however, requires recognizing the ways in which data-driven algorithmic systems specifically make determining these parameters challenging. We therefore update Nissenbaum's four barriers to accountability in relation to these systems, and clarify the ways that each barrier obscures and complicates realizing a moral, relational accountability framework in practice (Section G.3). Finally, we conclude by suggesting ways of weakening these barriers to accountability, thereby strengthening accountability practices for the entire field (Section G.4).

G.1.1 Technological Interventions in Accountability

Re-visiting the four barriers requires engaging with the significant body of work on accountability produced in the interim. Rather than comprehensively reviewing existing literature—an undertaking already addressed in, e.g., Wieringa [622] and Kohli et al. [323]—we highlight three areas of work that we find useful for our analysis:

Standards of care. These play a crucial role in building a culture of accountability — establishing best practices and formal guidelines for ensuring that concrete practices align with agreed-upon values (e.g., safety). In engineering, standards of care dictate the behaviors and outputs expected of sound work. For data-driven algorithmic systems in particular, they have taken the form of annotations [52], audits [16], and frameworks concerning the appropriate use of data and other artifacts, which are often developed and used in the production of AI/ML systems [83, 226, 287, 401, 416, 537]. Taken together, these standards of care support accountability by making the intentions and expectations around such systems concrete; they provide a baseline against which one can evaluate deviations from expected behavior and, accordingly, are used to review and contest the legitimacy of specific applications of data-driven techniques. Some scholars have re-framed such standards around harmed and vulnerable parties [410, 487]. This work makes clear that standards of care, while important for developing actionable notions of accountability, do not guarantee accountability on their own [172, 595]. Algorithmic impact assessments attempt to fill this gap [424]. They task practitioners with assessing new technologies in terms of their anticipated impacts [410, 529], and they formalize accountability relationships in ways that may systematically address and correct algorithmic harms.

Law and policy. Literature on data-driven algorithmic systems generally concerns AI/ML-related harms and corresponding interventions. Work on liability spans both anticipated harms related to new or forthcoming data-driven technology, including autonomous vehicles and robotics [6, 22, 143, 187, 566], and not-yet-legally-cognizable harms, such as unfair discrimination due to demographically-imbalanced, biased, or otherwise-discredited training

data [269, 449, 609], privacy violations [130, 160, 308], and manipulation [332]. Regulatory and administrative scholarship tends to analyze data-driven algorithmic systems in relation to legislation and policy that predates many AI/ML technological developments [168, 426, 508, 532, 598, 621]. That said, recent regulatory interventions, including GDPR (the nascent, yet wide-reaching data-privacy policy in the EU [257, 309, 607]) and the California Consumer Privacy Act of 2018 [48], which have also been applied to AI/ML systems, are increasingly represented within the law and policy literature.

Law and policy approaches tend to focus on transparency, which is of broad import in democratic governance and is intimately connected to accountability [425]. Transparency is necessary for identifying responsible parties (in order to attribute harms to those who are responsible for them), and necessary for identifying the sources of these harms and potential mitigations [172]. Work in this area spans a range of urgent concerns surrounding lack of transparency in data-driven algorithmic systems. These include the obfuscation of data provenance [367, 606], particularly caused by the concentration of data ownership within data brokers [198, 340, 633], and insufficient transparency of algorithms and models, which contributes to the inscrutability of automated decisions [133, 334, 354]. Critics have argued that outsourcing legal decisions to automated tools, particularly data-driven tools that obscure underlying decision logic, can create a crisis of legitimacy in democratic decision-making [98, 127, 426, 588].

Computer science. Research in AI/ML has increasingly treated accountability as a topic for scholarly inquiry. In updating Nissenbaum's barriers, we address cases in which researchers explicitly recognize the relationship between

their work and accountability [313] — namely, in auditing and transparency — and work on *robustness*, which we identify as having significant implications for accountability, even when this work itself does not explicitly make the connection. Recent work on audits underscores the importance of being able to analyze algorithmic outputs to detect and correct for the harm of unfair discrimination [9, 487]. Transparency tends to be treated as a property of models, particularly whether a model is interpretable or explainable to relevant stakeholders [58, 180, 211]. More recently, computational work has begun to take a more expansive view of transparency, applying it to other parts of the ML pipeline, such as problem formulation, data provenance, and model selection choices [145, 210, 333, 543, 545].

Lastly, often overlooked, *robustness* draws attention to whether a model behaves as expected under likely, unlikely, anomalous, or adversarial conditions. Designing for and evaluating robustness implicates accountability, as it requires researchers to define their expectations of model performance rigorously; this in turn encourages inquiry into how to prevent deviations from those expectations, and to identify (and ideally correct for) such deviations. Robustness thus encompasses work in AI/ML that aims to achieve theoretical guarantees in practice [402, 629, 641], and work that, even in the absence of such guarantees, produces models with reproducible empirical behavior [80, 484]. Robustness also includes the ability for models to generalize beyond the data on which they were trained [284, 442], ranging from natural cases of distribution shift [321, 456] to handling the presence of adversaries that are trying to game model outputs [241, 460, 568].

G.2 Conceptual Framing

The conceptual framing of accountability for this paper draws from two sources of scholarship: 1) moral philosophy, which construes accountability as a relationship between and among multiple actors; and 2) political theory and the social sciences, largely focusing on work by Mark Bovens, whose framework for identifying accountability relationships has been particularly influential in contemporary scholarship on "algorithmic accountability"[4] [303, 333, 622].

G.2.1 Accountability in Moral Philosophy

Numerous efforts in moral philosophy have sought to develop a rigorous conception of accountability. We focus on two threads in the literature, *blameworthiness* and *relationships between moral actors*, and correspondences between the two.

Blame. Nissenbaum [445] anticipated a problem of diminishing accountability as societies become increasingly dependent on computerized systems. She attributed this likelihood to the emergence of barriers to accountability in computerized society, and turned to philosopher Joel Feinberg's work to explain how and why these barriers are prone to arise: Blame, defined in terms of *causation* and *faultiness*, is assigned to moral agents for harms they have caused due to faulty actions [200, 201].[5]

Following Feinberg, Nissenbaum conceives of actors as accountable when

[4] We discuss concerns with this phrase in Section G.3.3 (*scapegoat*).

[5] Neither of these elements is straightforward — in fact, they are both the subjects of centuries of philosophical and legal thinking. Faultiness, e.g., presumes free agency — a concept whose metaphysical character and role in moral attribution has been the subject of centuries' long debate — and is a basic concept in all legal systems that informs judgements of legal liability (categorizing harmful actions as intentional, reckless, and negligent) [199, 200].

they step forward to answer for harms for which they are blameworthy. Her concern was that in computerized societies too many circumstances would arise where no one would step forward to acknowledge blame for harm, whether due to genuine puzzlement or intentional avoidance. Accordingly, the barriers to accountability that she identifies arise because the conditions of accountability are systematically obscured, due, at some times, to circumstances surrounding computerization and, at other times, to a societal breakdown in confronting willful failures. *Many hands* obscures lines of causal responsibility (Section G.3.1); *"bugs"* obscures the classification of errors as instances of faulty action (Section G.3.2); *scapegoating computers* obscures answerable moral actors by misleadingly or mistakenly attributing moral agency to non-moral causes (Section G.3.3); and *ownership without liability* bluntly severs accountability from blame (Section G.3.4).

Relationality. An alternative conception of accountability expands the focus to consider responsibility in light of the relationships between moral actors. Watson [615], for example, argues that responsibility should cover more than *attributablility*, a property assigned to an actor for bringing about a given outcome [569]. A second dimension, which he calls *accountability*, situates responsibility in a *relationship among actors*. For Watson, "Holding people responsible is not just a matter of the relation of an individual to her behavior; it also involves a social setting in which we demand (require) certain conduct from one another and respond adversely to another's failures to comply with these demands" [615, p. 229]. Other work, including T.M. Scanlon's theory of responsibility, provides accounts of both *being responsible* and *being held responsible*, where the latter describes situations when parties violate relationship-defined norms [517, 538]. Accordingly, the characteristics of a harmed party might dictate whether, or

what, accountability is needed. For instance, if one causes harm in self defense, there may be no moral imperative to hold them accountable.

This work attempts to situate accountability in the social, political, institutional, and interpersonal relationships in which we are enmeshed. Accordingly, the relationship-defined obligations we have to one another — as spouses, citizens, employees, friends, etc. — may dictate what it is we are responsible for, as well as the types and degrees of accountability we can expect. By situating accountability not just as *attributability* between action and actor, but instead within a social framework, some of what has come out of the so-called "narrow" notion of accountability in political theory (discussed below in Section G.2.2) can be derived from the vantage of a more "pure" moral philosophy. Rather than formally pursuing this derivation here, we instead simply suggest that these notions of accountability need not be framed as alternatives to one another. Moral philosophy offers concepts through which a given relational framing — be it interpersonal, institutional, or political — can be said to be legitimate and ethically viable. Similarly, for practitioners holding a variety of organizational positions (in relation to one another), the moral responsibilities that individuals hold can shape the ethical obligations and specific forms of accountability at play.

G.2.2 Accountability in Political Theory and the Social Sciences

The work in moral philosophy discussed above aligns with work on accountability as a property of social structures [223], which holds it to be relational —

not merely as a requirement on an accountable party to "own up" to blameworthy action as an obligation *to* another. In the past few years, "algorithmic accountability" has attracted growing interest in approaches that are institutional or structural in character.

The work of political scientist Mark Bovens, particularly what he has labeled, a "narrow definition" [81, 82], has informed recent literature on accountability for "algorithmic systems" [622]. Prompted by a concern that newly formed governmental structures and public authorities in the European Union lack "appropriate accountability regimes" [82, p. 447], Bovens proposed that accountability obtains between two key roles: an *accountable actor* and a *forum*. Under certain conditions, or in the wake of certain incidents, accountability exists when an accountable actor has an enforceable obligation to a forum to explain and justify itself — to address a forum's questions and judgments and possibly suffer sanctions. Bovens calls this a "relational" definition because it locates accountability in a social relation between those occupying one role (e.g. governmental department, a public authority, or a person acting in an official capacity) and another (e.g., a different governmental entity, oversight committee, or even an individual acting in a relevant capacity, e.g. journalist). We read Bovens as gesturing toward four key parameters in any relational accountability framework for which appropriate values need to be specified:

Who is accountable?: *Accountable actors* may include those who are not directly responsible for harm (e.g., engineers) but are designated as accountable (or liable) because of their deep pockets, capacities to render explanations, or positions in organizational hierarchies, such as corporate officers or government procurers of data-driven systems.

For what?: Beyond legally-cognizable harms (e.g., bodily injury, property damage, pecuniary losses), harms particularly associated with data-driven algorithmic systems include privacy violations [130], automation-driven unfair discrimination [16], autonomy losses due to manipulation [12, 332], and any number of emergent harms associated with novel technologies and their deployment.

To whom?: The members of the *forum* may not just include those who are themselves harmed (or placed in harm's way through heightened risk). They may also include those deputized to represent and advocate on behalf of vulnerable parties, such as lawyers and public or special interest advocacy groups. Beyond direct advocates, these may include groups and individuals in oversight capacities such as journalists, elected officials, government agencies, professional societies, or the many publics which coalesce around particular matters of concern [409].

Under which circumstances?: This concerns the nature of the obligation — what *accountable actors* may owe to the forum (to explain, be judged, and address questions and challenges). For example, Moss et al. [424] describes an array of components that constitute accountability within impact assessment frameworks, noting that the specific obligations an actor owes to a forum depend on the norms of that relationship.

Bringing together the moral and the relational. Proponents of Bovens's relational framework claim that it illuminates the sociopolitical stakes of transparency and explainability, showing why these concepts are necessary for any accountability framework for data-driven algorithmic societies, even though they are ultimately not sufficient to constitute accountability in and of them-

selves [622]. Moreover, by defining actors' roles and capacities in terms of the respective sociopolitical structures in which we live, Bovens's framework is *not* directed at the rights and obligations we have to one another as bare moral actors. We note that bringing together Bovens's relational definition with the moral conception of accountability can help clarify the scope of possible values for the framework's parameters: those who have caused or contributed to harm through faulty action are contenders for the class of *accountable actors*, and those who have suffered harm (and/or their representatives) deserve a place among the members of the *forum*.

This point shows a confluence between accountability as answerability for blameworthy action, and accountability as a social arrangement. Being blameworthy for harm is (almost always) a sufficient condition for being designated an accountable actor; being harmed through blameworthy action is (almost always) a sufficient condition for being designated a member of the forum, empowered to demand explanations. These two conceptions of accountability — the moral and the relational — do not stand against one another as alternative solutions to the same problem; they are solutions to different problems that intersect in constructive ways.

Nevertheless, hard work remains to explain and justify concrete, appropriate values for these parameters, and to construct pervasive structures for accountability through context-bound contestation [410]. In Section G.3 below, we demonstrate how data-driven algorithmic systems heighten the barriers to accountability by further obscuring conditions of responsibility and fault, which in turn presents challenges for instantiating the four parameters of a moral, relational accountability framework.

G.3 Revisiting the Four Barriers to Accountability

In a typical scenario in which software is integrated into a functional system — fully or partially displacing groups of human actors — accountability could be displaced along with human actors who are its bearers. The cumulative effect of such displacements is the increasing incidence of harmful outcomes for which no one answers, whether these outcomes are major or minor, immediate or long-term, or accrue to individuals or to societies. Resuscitating accountability is no simple task because computerization sets up particularly troublesome barriers to accountability: *Many hands* (G.3.1), *"Bugs"* (G.3.2), *The computer as scapegoat* (G.3.3), and *Ownership without liability* (G.3.4) [445]. These interdependent barriers are not necessarily an essential quality of computer software. Rather, they are a consequence of how software is produced, integrated into institutions, and embedded within physical systems; they are a function of the wonderment and mystique that has grown around computerization, and the prevailing political economy within which the computer and information industries have thrived. In the sections that follow, we revisit the barriers to accountability with an eye turned toward their implications amid the massive growth and adoption of data-driven algorithmic technologies. We provide examples of the barriers in action and defer discussion of how the barriers can be weakened to Section G.4.

G.3.1 The Problem of *Many Hands*

The barrier of *many hands* arises due to the large number of actors often involved in the design, development, and deployment of complex computerized systems.

When such systems cause harm, it may be difficult to isolate the component(s) at its source and the agents responsible: "Where a mishap is the work of 'many hands,' it may not be obvious who is to blame because frequently its most salient and immediate causal antecedents do not converge with its locus of decision making" [445, p. 29]. Nissenbaum further analyzes the difficulty of *many hands* by showing how it operates at four different levels: 1) software is produced in institutional, often corporate, settings in which there is no actor responsible for all development decisions; 2) within these settings, multiple, diffuse groups of engineers contribute to different segments or modules of the overall deployed system, which additionally often depends on software implemented by other actors (in today's landscape, this may result in licensed or freely-available open-source software); 3) individual software systems often interact with or depend on other software systems, which themselves may be unreliable or present interoperability issues; 4) hardware, not just software, often contributes to overall system function, particularly in cyber-physical systems, and it can be difficult to pinpoint if harms occur due to issues with the code, the physical machine, or the interface between the two. Any and all of these four levels of *many hands* problems can operate simultaneously, further obscuring the source of blame.

These difficulties at the heart of the *many hands* problem persist, further complicated in numerous ways now that computer systems are ubiquitous rather than merely ascendant. We focus on how data-driven algorithmic systems complicate this barrier with novel challenges using two illustrative (though necessarily non-exhaustive) examples: 1) The *ML pipeline* — the multi-stage process by which machine-learned models are designed, trained, evaluated, deployed, and monitored; 2) Reliance of contemporary data-driven algorithmic systems on the *composability of openly-available ML toolkits and benchmarking suites*; these

toolkits, often developed and maintained by large tech companies, tend to be advertised as general- or multi-purpose, and are frequently (mis)used in specific, narrow applications.

The ML pipeline. The ML pipeline is a dynamic series of steps, each of which can involve multiple groups of actors, including designers, engineers, managers, researchers, and data scientists. The pipeline typically starts with problem formulation and, in commercial settings, results in the deployment and continued monitoring of a trained model [462]. Problem formulation involves the collection, selection, or curation of a dataset, followed by the operationalization of a concrete task to learn, such as classifying loan-granting decisions or generating natural-language text. The actors responsible for formulation may hand off their work to others responsible for implementation — choosing the type of model and the learning procedure to use for model training. In selecting the type of model, these actors may custom-design their own model architecture, or may defer to a pre-existing one, such as an off-the-shelf neural network, which has been designed by others, possibly at another company or institution.

Thereafter, training and evaluation begin, in which a group of developers run training many times, perhaps with multiple combinations of model types, training procedures, and hyperparameter values. These developers compare trained models, from which they select some "best"-performing model (or ensemble of models), where "best" is informed by a quantitative metric they have adopted, such as mean overall test accuracy. These stages, from formulation to evaluation, are often repeated dynamically: until the model passes the threshold of developer-specified performance criteria, the process can cycle from remodeling to tuning. If the model is deployed in practice, there is yet another set

of actors who monitor the model's ongoing behavior, ensuring that its behavior aligns with expectations developed during training and evaluation.

Each stage of the ML pipeline involves numerous actors — in fact, potentially *indefinitely* many actors if the pipeline employs third-party model architectures or ML toolkits, which we discuss below.[6] Thus, in practice, if a trained model causes harms, it can be extremely challenging to tease out particular actors who should answer for them. For example, harms could originate from how actors operationalize the learning task at the start of the pipeline [136], move from high-level abstraction to concrete implementation [530], or select hyperparameters or random seeds during model selection [145, 210, 543]. Blame could lie with actors in any part of the pipeline, or some combination thereof whose faulty actions may have been causally responsible for harm. Bias, for example, could creep in early, from the choice of dataset, and accumulate and become magnified further downstream during model selection. In other words, the diffuse and dynamic nature of the pipeline makes locating accountability extremely challenging. This can be understood as an issue of transparency — beyond the specific the problem of model interpretability — concerning *who* is responsible *for what*, and *how* this can be related to overarching accountability with respect to a model's ultimate use in practice [333].[7]

Multi-purpose toolkits. Practitioners and researchers often do not code model architectures or optimization algorithms from scratch. Just as Nissenbaum highlighted the integration of third-party software modules as the indefinite expansion of *many hands*, we note here that builders of data-driven algorithmic sys-

[6]Participatory design further expands the set of *many hands* to end-user stakeholders [545], illustrating an additional manifestation of the barrier: when harms occur, it is possible to shift blame to harmed end-users who were involved in the ML pipeline.

[7]This indicates why transparency in the form of model interpretability may be important, but is ultimately not sufficient, for identifying actors accountable for harms.

tems often rely on toolkits produced by others. To decrease the amount of time and money spent iterating the ML pipeline, these actors depend on the investment of tech companies with vast resources and large, concentrated pools of technical talent to develop and release efficient, correct, comprehensive, and user-friendly libraries of algorithm implementations, model architectures, and benchmark datasets [4, 398, 463].

Unlike more traditional modules, which only tend to contain reusable software algorithms, ML toolkits often also include large-scale, pre-trained models. Large companies train and release such models, like BERT [171], which smaller companies and individuals can use out-of-the-box or fine-tune for particular use cases. Since these pre-trained models are often intended for downstream use by users different from their developers, they are designed for a multiplicity of applications (i.e., to be general-purpose). However, users employ pre-trained models in specific domains; there is a gap between general design goals and specific deployment intentions, which has been shown can bring about bias-related harms. Determining blame for these types of harms is far from simple. For example, if intended use is under-specified, blame could lie at least partially with the pre-trained model's creator. Compounding this problem is the fact that ML presents a recursive turn in the *many hands* problem Nissenbaum highlighted, in that many ML systems incorporate pre-trained components that are, themselves, the product of *many hands*. Nevertheless, tracing such harms presents an addressable technical challenge, not an insurmountable epistemological barrier.

In relation to a moral, relational accountability framework, this barrier obscures ...

Who is accountable: *Many hands* is central to identifying an accountable actor within Bovens's framework [81]. This problem has long characterized challenges in holding corporate actors, institutions, and organizations accountable, and while it certainly constituted a barrier to accountability in 1996 [445], it has only become more difficult to understand who is accountable in a data-driven algorithmic society. Code reuse — taken as a virtue in software development — has now been extended to model reuse, in turn generating a host of problems for equity and reliability by making it difficult to identify all the actors who contributed to components of an ML pipeline. Knowing who is responsible for these components as they are repurposed, as well as who ought to be responsible for incorporating those components into a downstream system, becomes prohibitively difficult for a forum to ascertain on its own, let alone for it to demand any explanations or changes in actors' behavior.

For what: The problem of *many hands* extends the above question to determining *what* an actor might be accountable for in relation to harms, in that it is hard to isolate which part of an ML pipeline actually contributes to an error or harm. Repurposed models may introduce dataset imbalances and proxies for protected categories without adequate scrutiny (or even the opportunity for scrutiny) by those assembling downstream components of a system. This raises questions of appropriate use, wherein it is difficult to tease apart the responsibility of those who produced a component to adequately stipulate the limits of its appropriate use and the responsibility of those who use that component to ensure it is appropriate for the uses to which they are putting it.

To whom: *Many hands* is primarily a barrier to knowing *who* is accountable, but it is also a barrier to knowing *to whom* those accountable actors are account-

able where, for example, a differential error rate may exist for some population $\mathcal{P}$, but a specific harm occurs for an individual $p \in \mathcal{P}$. In such a case, it is difficult to determine whether accountability ought to be rendered to $\mathcal{P}$, because of the heightened risk of harm to which the entire population has been exposed, or only to p, who suffered harm because of their membership in $\mathcal{P}$. This is a *many hands* problem because of the difficulty in knowing where within the ML pipeline risk was produced for the group, e.g., through training or dataset imbalances, and where it was produced for individuals, e.g., through implementation choices.

Under which circumstances: The problem of *Many hands* presents a barrier even when standards of care exist, as it is difficult for actors to know precisely whom they should exercise that care toward (see "to whom" above). Standards of care, which are grounded in normative assumptions about appropriate component (re)use, are less straightforward to develop where *many hands* are involved, as social practices which link actors together are obscured throughout the ML pipeline.

G.3.2 *"Bugs"*

Nissenbaum uses the term *"bug"* to cover a variety of issues common to software, including "modeling, design, and coding errors." *"Bugs"* are said to be "inevitable," "pervasive," and "endemic to programming," "natural hazards of any substantial system" [445, p. 32]. Even with software debuggers and verification tools that can assure correctness, *"bugs"* emerge and cause unpredictable behavior when software systems are deployed and integrated with each other

in the real world [388, 547]. The rhetorical power of *"bugs"* is that they are predictable in their unpredictability; they serve as a barrier to accountability because they cannot be helped (except in obvious cases), and therefore are often treated as an accepted "consequence of a glorious technology for which we hold no one accountable" [445, p. 34].

What we consider to be the "inevitable" can change over time as technology evolves, with certain types of *"bugs"* spilling over into the avoidable. For example, evolving norms and new debugging tools can rebrand the "inevitable" to be sloppy or negligent implementation, at which point programmers can be held to account for such errors. Similarly, the advent of data-driven algorithmic systems has indicated that this malleability also extends in the other direction: new technological capabilities can both contract and expand what we consider "inevitable" *"buggy"* behavior. That is, while these systems contain *"bugs"* of the "modeling, design, and coding" varieties that Nissenbaum describes for rule-based programs, the statistical nature of data-driven systems presents additional types of harm-inducing errors, which may present an additional barrier to accountability.[8] Where misclassifications, statistical error, and nondeterministic outputs cause harm — and are presented as inevitable and unavoidable — may impede the attribution of blame.

In 1996, it may have been evident that labeling certain errors as *"bugs"* was a mere ploy to dodge blame. Today, certain types of errors are more plausibly asserted to be an inherent part of ML, attributable to its statistical nature. Misclassification, statistical error, and nondeterminism seem to turn the notion of *"bug"* on its head: indeed, many experts would as readily call these *features* of machine

[8]Of course, statistical software is not new to ML; however, the proliferation of data-driven algorithmic systems has clarified the prevalence of such errors.

learning, not *"bugs"*.[9] Nevertheless, regardless of where one attempts to draw the line, these errors share common elements with the *"bugs"* Nissenbaum describes — namely, they undermine our ability to reason, conclusively, about causality and fault. Insofar as they are accepted as an "inevitable," "pervasive," and "consequence of a glorious technology," they constitute a barrier to accountability [445]. Below, we illustrate this point with concrete instances of *"bugs"* deemed unavoidable in data-driven algorithmic systems.

Faulty modeling premises. As discussed in Section G.3.1, data-driven algorithmic systems require significant modeling decisions prior to implementation. For example, choosing a model to learn necessarily involves abstraction and can have significant ramifications [462, 530]. Assumptions during this stage of the ML pipeline can bias the resulting computational solution space in a particular direction [215], for example, assuming a linear model is sufficient to capture patterns in data precludes the possibility of modeling non-linearities. When such biases involve over-simplified or faulty reasoning, they can result in model mis-specification and the introduction of "modeling error bugs." Such mis-specifications may include the assumption that values like fairness and accuracy are correctly modeled as a trade-off to be optimized [136], and that physical characteristics can serve as legitimate classification signals for identifying criminals [625] or inferring sexual orientation [561, 614]. More generally, a common modeling error may arise from assuming, in the first place, that a problem is amenable to classification — that it is possible to divide data examples into separable categories [546, 563]. Even if it is possible to train mis-specified models like these to behave "accurately" (i.e., to return better-than-chance results after learning these tasks), conclusions drawn from false premises will be

[9] We return to this in Section G.3.3 (*scapegoat*) and is why we leave *"bugs"* in quotes.

unsound [136]. If modeling assumptions are unclear or elided, an actor may evade accountability by blaming inexplicable, unavoidable *"bugs"* endemic to computer software instead of taking responsibility for otherwise opaque errors.

Individual errors. Even if one's premises are not faulty, the ML pipeline can still produce models that cause harm. Trained ML models exhibit errors that can harm individuals if their effects, for example, violate privacy or cause manipulation [203, 332, 381, 432]. ML has several techniques to quantify and minimize error [74, 260], and yet even the most robust, well-trained models report imperfect accuracy. In fact, a model that achieves 100% accuracy is usually considered suspect, likely over-fit to the training data and to exhibit poor performance when presented with new examples [262, 491, 557]. Therefore, when individual errors occur, they can be treated as inevitable, just like the *"bugs"* Nissenbaum describes, displacing responsibility for the harms such errors cause affected individuals.

Bad model performance. Unexpectedly bad overall model performance can likewise be excused as a *"bug,"* rather than a blameworthy error. Consider a hypothetical example of a (well-formulated) computer vision system used to detect skin cancer, whose training and evaluation indicate will have an accuracy rate of 94%. Once deployed, if the model coheres with (or even out-performs) its promised performance, then developers can claim that any mis-classifications were expected.[10] Since expected accuracy is a probabilistic claim about what is likely to occur, deviations from expectation can and do occur. When monitoring a deployed model, over time, if this deviation yields a substantial decrease in expected accuracy, developers may dodge accountability by ascribing the fail-

[10]Individual errors can pose additional challenges for accountability: the model may still overall exhibit an expected degree of error (i.e., be within a margin of error), for which it is possible to scapegoat the statistical nature of ML (Section G.3.3).

ure to the amorphous category of *"bug"*, instead of admitting that it resulted from human negligence, poor generalization, distribution shift, or other faulty behavior.

In relation to a moral, relational accountability framework, this barrier obscures . . .

Who is accountable: Accountability for *"bugs,"* even within the expanded definition of *"bugs"* provided above, emerges from specific regulatory regimes, corporate compliance practices, and contracting relationships. Civil law has a crucial role in determining the relationship between forums and responsibilized actors, which is often inflected by those who have the capacity to intervene or have benefitted from a particular action. *"Bugs"* present a particular challenge to determining who is accountable when they are seen as endemic to ML, or as produced by non-determinism inherent to the domain in which an algorithmic system is deployed (a challenge shared by the *scapegoating* barrier).

For what: *"Bugs"* remain a barrier to accountability because of the difficulty they pose to actors and forums trying to specify whether individual errors, bad model performance, faulty assumptions, or other mistakes contributed to a harm.

To whom: *"Bugs"* may affect an entire class of individuals, a community, or all of society, but evidence of harm may only accrue at the level of a specific individual, presenting a barrier for actors and forums interested in knowing to whom accountability ought to be rendered.

Under which circumstances: Algorithmic systems inevitably rely on some de-

gree of abstraction and make specific assumptions about the underlying nature of the phenomena they model [136, 530]. Under circumstances of imperfect information about every possible aspect of a data-driven algorithmic system (which is most of the circumstances outside the lab), *"bugs"* of the character described above may exist and contribute to this barrier to accountability.

G.3.3 *The Computer as Scapegoat*

Blaming a computer may pose a barrier to accountability, because "having found one explanation for an error or injury, the further role and responsibility of human agents tend to be underestimated" [445, p. 34][187]. To explain why people could plausibly blame computers for wrongdoing, Nissenbaum cites the role computers may play in "tasks previously performed by humans in positions of responsibility;" whereas before the human would be indicated as the blameworthy party, the computer has now taken up that role. And yet, even as computer systems have become immediate causal antecedents to an increasing number of harms, they lack moral agency and thus cannot be the bearers of moral blame [445]. In this section, we discuss how *scapegoating the computer* has become even more complicated in the landscape of ubiquitous data-driven algorithmic systems. In the examples below, the system is made to bear the sins of the responsible party, the individual or the institution that has agency and is capable of carrying moral blame.

Moral agency. As data-driven algorithmic systems have become pervasive in life-critical contexts, there has been a corresponding tendency to anthropomorphize and view technological processes as akin to human cogni-

tion [64, 475, 586]. These systems are described by their developers and commentators as intelligent, implying that they have agency as autonomous actors and thus rhetorically positioning them as blameworthy for error. However, directing blame toward data-driven algorithmic systems effectively imbues them with moral agency, ascribing them the ability to act intentionally [519]. Nissenbaum likens blaming a computer to blaming a bullet in a shooting: While the bullet can be said to play an active, causal role, it cannot be said to have been intentional in its behavior. In the same vein, a data-driven algorithmic system may play a central role in life-critical decisions, and may even be said to *make a choice* in a particular task, but a choice lacking deliberate intention, a precondition for moral agency [519].[11]

"Accountable algorithms". This popular banner-phrase makes *algorithms* the subject of accountability [334], even though algorithms are not bearers of moral agency and, by extension, moral responsibility. It places responsibility on technology, not its developers, owners or operators, and it reduces accountability to a piecemeal, procedural quality that can be inferred from technology, rather than a normative concept that has to do with the moral obligations that people have toward one another. The phrase further occludes proper attribution of accountability by fixating attention on algorithms rather than on systems that are deployed in practice, within and through which algorithms function [144]. When, for example, studies of fairness in AI/ML-assisted judicial bail decisions fixate on respective algorithms, they fail to capture key inequities that are systemic in complex sociotechnical systems, of which AI/ML techniques are just one part [41].

[11]This is consistent with scholarship in legal theory concerning AI, algorithms, agency, and personhood [34, 64, 93, 97, 276, 358, 596].

Mathematical guarantees. Directing blame away from people and corporations can be either strategic or inadvertent. In some cases, a group of harmed individuals does not know whom to blame (*many hands*) and settles on blaming the system. In others, scapegoating the system can be a way by which a moral actor dodges and dissipates public ire, for example, in the now-canonical example of Northpointe exhibiting bias in its risk-assessment tool [26]. Rather than attributing this bias to a mistake or *"bug,"* Northpointe blamed the fundamental incompatibility of different algorithmic operationalizations of fairness as the source of the problem (and pointed to a specific measure, for which bias was not detectable, as evidence of blamelessness). Reliance on mathematical guarantees can reinforce barriers to accountability and divert attention away from its appropriate subjects. One can see this when a given system has a theoretically-guaranteed (and empirically-verified) upper bound on its error. If the system behaves within its guaranteed margin of error, it becomes possible to treat that margin as an immutable attribute of the system (rather than, more appropriately, the result of human-made decisions), and to scapegoat the system for any particular errors that fall within this margin.

Let us consider the same case we discussed for the problem of individual errors in *"Bugs:"* The engineers show that a system is 94% accurate for tumor detection, and validate that this is in fact the case in practice. Above, we talked about this example in terms of individual errors, for which responsibility for harm could be excused due to *"buggy"* behavior. Rather than analyzing behavior at this level of individual decisions, one can also examine the behavior *of the model overall*. If the frequency of mis-classifications is within the model's guaranteed error rate, the engineers could attempt to excuse all resulting harms by gesturing to the fact that the model is performing *exactly as expected*. In short,

satisfying mathematical guarantees can serve as a *scapegoat* because pointing to mathematical claims satisfied at the model-level can serve to obscure the need to account for harms that occur at the individual-decision level.[12]

Non-determinism. When data-driven algorithmic systems err, their errors can be attributed to the stochastic or otherwise non-deterministic components of either the system itself or the phenomena the system is modeling. In particular, systems that involve ML involve randomization, for example, by shuffling the order in which training data examples are presented to an algorithm. While such features of ML algorithms may seem like technical minutiae, in fact, they introduce stochasticity into the outputs of machine-learned models: training the same model architecture on the same dataset with the same algorithm — but changing the order in which the training data are supplied to the algorithm — can yield models that behave differently in practice. For example, as Forde at al. [210] shows, changing the order that the data examples are presented to train a tumor-detection model can lead to surprisingly variable performance. The relationship between training-data-ordering and resulting variance in model performance is under-explored in the technical literature. Thus, such differences in model performance are often attributed to an inherent stochasticity in ML. The randomization used in ML systems — randomization on which these systems depend — becomes a *scapegoat* for the harms it may cause, such as missed tumor detection. In attributing the harms to mathematical chance, attention is drawn away from appropriate accountable agents.

[12]One could see-saw back-and-forth between *"bug"* and *scapegoat* to evade accountability. If satisfying guarantees at the overall model-level is rejected as a rationale for an individual harm, one could claim there is a *"bug;"* if calling an individual decision *"buggy"* is rejected, and the model is classifying within its expected error, one could then displace blame by arguing that the model is behaving according to its specification.

In relation to a moral, relational accountability framework, this barrier obscures . . .

Who is accountable: Similar actors are accountable as those described in *"Bugs,"* although the barrier presented by *scapegoating* is embedded in its implicit suggestion that entities are accountable, rather than those who are the responsible actors (see the nebulousness of *many hands*), or that no responsible actor can be found because a harm occurred through randomness or chance.

For what: *Scapegoating* produces barriers to understanding the *for what* of accountability in identical ways as described above in *"Bugs,"*. It also contributes an additional difficulty when mathematical guarantees are offered that allow for some minimal degree of undesirable behavior in a system, or the system is characterized as non-deterministic in ways that would indemnify otherwise responsibilized actors from accountability for outcomes stemming from such undesirable behaviors.

To whom and *under which circumstances*: Same as in Section G.3.2.

G.3.4 Ownership without Liability

Nissenbaum [445] highlights a dual trend in the computer industry: 1) strengthening property rights and 2) avoiding liability. Behavioral trends that informed these assertions have persisted in the decades since, with lively public debates over the fit of traditional forms of intellectual property (i.e., copyrights, patents, and trade secrets) to digital products such as software, data, databases, and algorithms [133, 216], and subsequent expensive legal struggles among indus-

try titans [71]. Similarly, we have seen explicit denials of liability expressed in shrink-wrap licenses, carried over into so-called "click-wrap" licenses, and Terms of Service disclaimers accompanying websites, web-based services, mobile apps, Internet of Things devices, content moderation decisions, and the like [130, 329, 366, 573].

Before addressing how we see these trends carry forward in the contemporary landscape, we need to qualify our observations. Property and liability are weighty legal concepts with long histories and rich meanings. Narrowing our view to digital technologies, even before Nissenbaum [445], a robust literature had grown over questions of ownership — questions that have persisted through numerous landmark court cases. Liability, too, is a core legal concept that is increasingly an issue in relation to the products and services of digital industries. It lies outside the scope of this paper to attempt meaningful insights into these concepts as they manifest in scholarship, law, and the courts. However, it is useful to observe broad patterns and anticipate the likely actions of stakeholders. For a start it is not difficult to see how the trends toward strong ownership and weak liability reinforce barriers to accountability, and also to understand why industry incumbents might support them: liability is costly and strong property rights enrich rights holders and empower them against competitors. Four lines of advocacy on behalf of industry interests are noted below, supplementary to those discussed in Nissenbaum [445]:

- Third-party providers of data-driven algorithmic systems refuse to expose their systems to scrutiny by independent auditors on grounds of trade secrets [133, 216]. As long as experts maintain that transparency is necessary to evaluate the ML pipeline and AI development, strong property rights

that block scrutiny are barriers to accountability.

- Manufacturers and owners of cyber-physical systems, such as robots, Internet of Things devices, drones, and autonomous vehicles, evade liability for harms by shifting blame to environmental factors or humans-in-the-loop [358]. In this respect, the barrier of *ownership without liability* for data-driven algorithmic systems suggests a twist on the problem of *scapegoating* (Section G.3.3): treating "the human user as scapegoat" — claiming the user has mis-used an AI- or ML-enabled system in order to obscure responsibility for unclear, under-specified, or deliberately misleading user interfaces or expected use, as has happened with Tesla and accidents concerning its (so-called) "AutoPilot" autonomous driving feature [78].
- Almost without question the computer industry, having metamorphosed into the data industry, has assumed ownership over data passing through its servers [198, 340, 449]. We still do not have clear rules of liability for industry actors when their servers, holding unimaginable quantities of data, are breached [535]. Nor do we have sufficient insight into the completeness, quality, or validity of data, or the means to hold anyone liable for its misuse.
- Technology companies hold unprecedented sway over regulation. Twenty-five years ago, the software industry was already a force to be reckoned with and successfully persuaded Congress that imposing legal constraints would stifle innovation — that societal well-being depended on a nascent industry that could not flourish under excessive regulatory and legal burden.

In relation to a moral, relational accountability framework, this barrier ob-

scures . . .

Who is accountable: Having already enumerated above the many difficulties these barriers pose for tracing relationships of accountability, they generally pertain to the problem of *ownership without liability*, as well. Additionally, questions of how liability is adjudicated in practice may obscure who is liable, what kind of liability they hold, or what they are liable for, while leaving intact the ways in which the benefits of data-driven algorithmic systems accrue to their developers, designers, and operators.

For what: *Ownership without liability* affects the very contours of what an actor can be found liable for. However, this does not absolve that actor of their moral responsibility or obviate the need for them to be held accountable for the consequences of their actions, the systems they oversee, or from which they benefit.

To whom: *Ownership without liability* is a barrier to accountability for those who may stand as plaintiffs in civil cases and representatives of those affected.

Under which circumstances: *Ownership without liability* is a barrier to accountability where those who suffer a harm lack standing in a court of law. This may be because a harm is not cognizable to courts (see, e.g., Metcalf et al. [409]), the harmed party does not constitute a certifiable class, or the nature of the harm is obscured through the ways harms are foisted onto *scapegoats* or dismissed as *"bugs"*.

G.4 Weakening the Barriers

Nissenbaum [445] warned of a waning culture of accountability — harms befalling individuals, groups, even societies, were being cast merely as sufferers' bad luck. In the previous section, we revisited the four barriers in light of data-driven algorithmic systems and found that the framework still provides a useful lens through which to locate sources contributing to the dissipation of accountability. Weakening the barriers would clear the way for more sound attribution of blame, in turn setting up a stronger societal expectation for blameworthy parties to step forward and take account. But we have also argued that accountability in algorithmic societies involves more: stepping forward is a necessary component of accountability, but it is insufficient (Section G.2). Because the barriers we have described may not all be weakened, even with a firm resolve to identify blameworthy parties, we need more than astute attention on a case-by-case basis. To build a lasting culture of accountability, a necessary supplement involves establishing persistent institutional frameworks for identifying accountable parties (i.e., individuals, groups, or organizations) and for calling them to answer. Simultaneously, such frameworks should invest others with the powers to call these parties to account.

Any technical interventions that the research community has already developed — notably, those that we have emphasized concerning transparency, audits, and robustness — would need to be folded into such a framework, and their use justified in these moral and relational terms. For example, any technical definition of transparency is unlikely to satisfy the needs of all those who comprise a forum and who may hold variable or inconsistent ideas about what it might mean for a model to be "interpretable." Technical assertions of robustness

say what expectations are, but leave unanswered the question of the conditions under which deviations from expectations ought to be expected or remedied.[13] Relational treatments of these issues, it would seem, require that the obligation be tuned to the various needs of all members of the forum.

Taking each barrier in turn. A moral and relational accountability framework opens the aperture to addressing *many hands* (Section G.3.1). In principle, many, if not all, of the *many hands* could be designated as accountable actors. Deliberate consideration of the *many hands* problem is clearly called for by those who develop licensing agreements relying on normative assumptions about appropriate use and reuse within the ML pipeline, and in articulating engineering best practices empirically against theoretical assumptions of robustness. This includes dataset creators, model developers, decision and control systems designers, vendors, and operators of these systems. Developing rigorous standards of care could help mitigate the problems of inappropriate use of pre-trained models and unclear measures of quality control at different stages of the ML pipeline. For example, robust auditing mechanisms at each stage, rather than approaching audit as an end-to-end concern [487], or worse, as a purely *post hoc* endeavor, could help clarify the relationship between stage-specific issues and resulting harms.

Addressing various harms, depending on how they are contextualized, can implicate either the barrier of *"bugs"* or *scapegoating the computer* (Sections G.3.2 & G.3.3). For example, we note that the computer science community could have either treated algorithmic harms due to unfair discrimination as a *"bug"* or blamed them on intrinsic aspects of AI/ML — and yet, it did not. In-

[13]Moreover, if assumptions underlying such assertions are voided when moving from theory to deployment, robustness estimates can degrade in practice.

stead, unfairness has more often been ascribed to biased or imbalanced training data [208, 307] — data that exhibits historical biases that are arguably "pervasive" and "unavoidable." This community could have pursued some "tolerable" degree of unfavorable outcomes in the real world (ideally, in consultation with those adversely impacted), and developed ways of ensuring models met that more "tolerable" specification, under specific conditions. This, notably, would still have allowed developers to evade accountability by *scapegoating* inherent properties of the model as instead deserving of blame.

However, instead of treating unfairness as an aspect of accountability, much technical work on algorithmic fairness has attempted to address unfairness harms by developing training algorithms that are robust to biased input data. The field of algorithmic fairness therefore serves as an example that challenges the narrative of the invulnerability of the barriers. The technical community and its interlocutors have demanded more from ML modelers concerning the treatment of unfair discrimination. The community has set expectations concerning the necessity of interventions to root out and correct for unfairness, thereby weakening the barriers of *scapegoating* or being attributed to *"bugs"*. This example could, and we believe should, encourage similar treatment of other issues like robustness and its relationship to privacy violations, or adversarial ML and its relationship to manipulation.

Lastly, *being liable* is related but not identical to being accountable (Section G.3.4). The latter is applied to blameworthy parties who step forward to answer, the former to parties who step forward to compensate victims of harm. Often liability is assigned to those who are found to be blameworthy. If lines of accountability are blurred, for example, as a consequence of the barriers we

have discussed, harms due to AI/ML and other data-driven algorithmic systems will be viewed as unfortunate accidents; the cost of "bad luck" will settle on victims. Instead, legal systems have developed approaches, such as strict liability, to compensate victims harmed in certain types of incidents even without a demonstration of faulty behavior. Strict liability assigned to actors who are best positioned to prevent harm is sound policy as it motivates these actors to take extraordinary care with their products. Barriers such as *many hands* make the attribution of blame difficult. Strict liability for a range of harms that are produced by *many hands* would shift the "bad luck" from victims to those best positioned to mitigate and prevent such harms.

Eroding the barriers of accountability is a key societal challenge requiring multiple forms of expertise and, with respect to ML especially, the use of these tools needs to be justified. Just as mature political governance requires durable institutions and formal attributions of rights and duties, we have similar needs for the governance of producers, purveyors, and operators of data-driven algorithmic systems. That is, as we have contended throughout this paper, accountability is moral *and* relational. It depends on social, legal, and political structures that provide legitimacy for the checks actors and forums place on each others' behavior; it depends on the way those checks are internalized as professional, personal, legal, and ethical duties that motivate actors' personal responsibility. Multi- and inter-disciplinary research on accountability, fairness, and transparency — given its potential to bring together an array of expertise focused on themes of equity and justice — is uniquely positioned to help develop a moral, relational accountability framework. Such structures provide legitimacy, as well as the professional codes and standards of care, disciplinary

norms, and personal mores that tie moral and relational forms of accountability together. The future work of creating these structures, as noted earlier, is no small undertaking, it lies in the sociopolitical contestations, the hard, deliberative work of living within a pluralistic society, by the many constituencies implicated in any particular computational system.

G.5 Conclusion

In this paper we revisited Nissenbaum's "four barriers" to accountability, with attention to the contemporary moment in which data-driven algorithmic systems have become ubiquitous in consequential decision-making contexts. We have drawn on conceptual framing from Nissenbaum's use of the concept of *blameworthiness* and how it can be aligned with, rather than cast in opposition to, Bovens's work on accountability as a *relational property of social structures* [81, 82]. We have demonstrated how data-driven algorithmic systems heighten the barriers to accountability with regard to determining the conditions of blame, and have looked ahead to how one might endeavor to weaken the barriers. In particular, we have put forward the conditions necessary to satisfy a moral and relational accountability framework, discussed how the development of such a framework would weaken the barriers, and argued that an interdisciplinary and multidisciplinary research community is uniquely positioned to construct such a framework and to develop lines of inquiry to erode the barriers to accountability.

Given our tender historical moment, addressing why these or those parties belong in the forum or in the set of accountable actors, why those obligations

are justified, and, of course, evaluating the numerous permutations the relational nature of the approach demands is the provenance of future work. No easy formulations make sense until we have developed a rigorous approach to justification. In our view, this calls for expertise in relevant technologies, moral philosophy, the prevailing political economy of data and computing industries, organizational sociology, current political and regulatory contexts, domain area expertise, and more. It is not that all these are needed all the time; but any of them may be called in to develop linkages between proposed values and social welfare.

BIBLIOGRAPHY

[1] A & M Records, Inc. v. Abdallah, 1996. 948 F. Supp. 1449 (C.D. Cal. 1996).

[2] A & M Records, Inc. v. Napster, Inc., 2001. 239 F.3d 1004 (9th Cir. 2001).

[3] Daniel Abadi. Consistency Tradeoffs in Modern Distributed Database System Design: CAP is Only Part of the Story. *Computer*, 45(2):37–42, February 2012.

[4] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, page 265–283, USA, 2016. USENIX Association.

[5] ABKCO Music, Inc. v. Harrisongs Music, Ltd., 1983. 722 F.2d 988 (2d Cir. 1983).

[6] Kenneth S. Abraham and Robert L. Rabin. Automated Vehicles and Manufacturer Responsibility for Accidents: A New Legal Regime for a New Era. *Virginia Law Review*, 105:127–171, 2019.

[7] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning From Data: A Short Course*. AMLbook, USA, 2012.

[8] José A Adell and Pedro Jodrá. Exact Kolmogorov and total variation distances between some familiar discrete distributions. *Journal of Inequalities and Applications*, 2006(1):64307, 2006.

[9] Philip Adler, Casey Falk, Sorelle A. Friedler, Tionney Nix, Gabriel Rybeck, Carlos Scheidegger, Brandon Smith, and Suresh Venkatasubramanian. Auditing black-box models for indirect influence. *Knowledge and Information Systems*, 54:95–122, 2018.

[10] National Highway Traffic Safety Administration. Federal Automated Vehicles Policy: Accelerating the Next Revolution In Roadway Safety. Technical report, U.S. Department of Transportation, 2016.

[11] Adobe. Experience the Future of Photoshop With Generative Fill, July 2023. `https://helpx.adobe.com/photoshop/using/generative-fill.html`.

[12] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, and Hao Li. Protecting World Leaders Against Deep Fakes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, page 8, Long Beach, CA, June 2019. IEEE.

[13] Meta AI. Use Policy, 2023. `https://ai.meta.com/llama/use-policy/`.

[14] Stability AI. Stable Diffusion XL, 2023. `https://stability.ai/stablediffusion`.

[15] Shigeki Aida. Uniform positivity improving property, sobolev inequalities, and spectral gaps. *Journal of functional analysis*, 158(1):152–185, 1998.

[16] Ifeoma Ajunwa. An Auditing Imperative for Automated Hiring. *Harv. J.L. & Tech.*, 34, 2021.

[17] Fatih Kadir Akın. Awesome ChatGPT Prompts. *GitHub*, 2023. `https://github.com/f/awesome-chatgpt-prompts`.

[18] Dan Alistarh, Christopher De Sa, and Nikola Konstantinov. The Convergence of Stochastic Gradient Descent in Asynchronous Shared Memory. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, page 169–178, Egham, United Kingdom, 2018. PODC '18.

[19] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1709–1720, New York, NY, 2017. Curran Associates, Inc.

[20] Ryan Alweiss, Yang P Liu, and Mehtaab Sawhney. Discrepancy minimization via a self-balancing walk. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 14–20, 2021.

[21] Amazon. S3 Storage Classes, July 2021. `https://aws.amazon.com/s3/storage-classes/`.

[22] American Association for Justice (AAJ). Driven to Safety: Robot Cars and the Future of Liability, 2017.

[23] American Broadcasting v. Aereo, 2014. 134 S. Ct. 2498 (2014).

[24] Anderson v. Stability AI, Ltd., 2023. No. 3:23-cv-00201 (N.D. Cal. Jan. 13, 2023).

[25] Christophe Andrieu and Gareth O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725, 2009.

[26] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. *ProPublica*, 23(2016):139–159, May 2016.

[27] Anthropic. Introducing 100K Context Windows, May 2023. `https://www.anthropic.com/index/100k-context-windows/`.

[28] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. *Operating Systems: Three Easy Pieces*. Arpaci-Dusseau Books, 1.00 edition, August 2018.

[29] Associated Press v. Meltwater U.S. Holdings, Inc., 2013. 931 F. Supp. 2d 537 (S.D.N.Y. 2013).

[30] Mark Van Atten. *On Brouwer*. Cengage Learning, 2004.

[31] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[32] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, et al. Constitutional AI: Harmlessness from AI Feedback, 2022.

[33] Peter Bailis, Shivaram Venkataraman, Michael J. Franklin, Joseph M. Hellerstein, and Ion Stoica. Probabilistically Bounded Staleness for Practical Partial Quorums. *Proc. VLDB Endow.*, 5(8):776–787, April 2012.

[34] Jack M. Balkin. The Path of Robotics Law. *California Law Review Circuit*, 6:45–60, 2015.

[35] Jack M. Balkin. Free Speech in the Algorithmic Society: Big Data, Private Governance, and New School Speech Regulation. *UC Davis Law Review*, 51(3):1149–1210, 2018.

[36] Alexandru Baltag and Sonja Smets. Probabilistic dynamic belief revision. *Synthese*, 165(2):179–202, 2008.

[37] Derek Bambauer and Mihai Surdeanu. Authorbots. *Journal of Free Speech Law*, 3:375, 2023.

[38] Jane R. Bambauer, Tal Zarsky, and Jonathan Mayer. When a Small Change Makes a Big Difference: Algorithmic Fairness Among Similar Individuals. *UC Davis Law Review*, 55:2337–2419, 2022.

[39] Kenneth A. Bamberger. Technologies of Compliance: Risk and Regulation in a Digital Age. *Texas Law Review*, 88(4):669–740, 2010.

[40] Marco Banterle, Clara Grazian, Anthony Lee, and Christian P Robert. Accelerating Metropolis-Hastings algorithms by delayed acceptance. *Foundations of Data Science*, 1:103, 2019.

[41] Chelsea Barabas, Colin Doyle, JB Rubinovitz, and Karthik Dinakar. Studying up: reorienting the study of algorithmic fairness around issues of power. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 167–176, New York, NY, USA, 2020. ACM.

[42] D. Barbara and H. Garcia-Molina. The case for controlled inconsistency in replicated data. In *[1990] Proceedings. Workshop on the Management of Replicated Data*, pages 35–38, Houston, TX, USA, Nov 1990. IEEE.

[43] Rémi Bardenet, Arnaud Doucet, and Chris Holmes. On Markov chain Monte Carlo methods for tall data. *The Journal of Machine Learning Research*, 18(1):1515–1557, 2017.

[44] Rémi Bardenet, Arnaud Doucet, and Chris Holmes. Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In *International Conference on Machine Learning*, 2014.

[45] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning: Limitations and Opportunities*. fairmlbook.org, 2019. http://www.fairmlbook.org.

[46] Solon Barocas and Andrew D. Selbst. Big Data's Disparate Impact. *California Law Review*, 104(3):671–732, 2016.

[47] Alessandro Barp, Carl-Johann Simon-Gabriel, Mark Girolami, and Lester Mackey. Targeted separation and convergence with kernel discrepancies. *arXiv preprint arXiv:2209.12835*, 2022.

[48] Catherine Barrett. Are the EU GDPR and the California CCPA becoming the de facto global standards for data privacy and protection? *Scitech Lawyer*, 15(3):24–29, 2019.

[49] David Barstow. Artificial intelligence and software engineering. In *Exploring Artificial Intelligence*, pages 641–670. Elsevier, 1988.

[50] Romain Beaumont. LAION-5B: A New Era of Large-Scale Multi-Modal Datasets. *LAION Blog*, March 2022. https://laion.ai/blog/laion-5b/.

[51] Stas Bekman. The Technology Behind BLOOM Training. *HuggingFace*, July 2022. https://huggingface.co/blog/bloom-megatron-deepspeed.

[52] Elena Beretta, Antonio Vetrò, Bruno Lepri, and Juan Carlos De Martin. Detecting discriminatory risk through data annotation based on bayesian inferences. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 794–804, New York, NY, USA, 2021. ACM.

[53] James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.*, 13:281–305, February 2012.

[54] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc., 2011.

[55] Dimitri P. Bertsekas. Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization: A Survey. In *Optimization for Machine Learning*. The MIT Press, 2011.

[56] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, Wesam Manassra, Prafulla Dhariwal, Casey Chu, Yunxin Jiao, and Aditya Ramesh. Improving Image Generation with Better Captions, 2023.

[57] Guram Bezhanishvili and Wesley H. Holliday. A semantic hierarchy for intuitionistic logic. *Indagationes Mathematicae*, 30(3):403 – 469, 2019.

[58] Umang Bhatt, Javier Antorán, Yunfeng Zhang, Q. Vera Liao, Prasanna Sattigeri, Riccardo Fogliato, Gabrielle Melançon, Ranganath Krishnan, Jason Stanley, Omesh Tickoo, Lama Nachman, Rumi Chunara, Madhulika Srikumar, Adrian Weller, and Alice Xiang. Uncertainty as a Form of Transparency: Measuring, Communicating, and Using Uncertainty. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, page 401–413. Association for Computing Machinery, New York, NY, USA, 2021.

[59] Stella Biderman, Kieran Bicheno, and Leo Gao. Datasheet for the Pile, 2022.

[60] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR, 2023.

[61] Joris Bierkens, Paul Fearnhead, Gareth Roberts, et al. The zig-zag process and super-efficient sampling for Bayesian analysis of big data. *The Annals of Statistics*, 47(3):1288–1320, 2019.

[62] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. *arXiv preprint arXiv:1801.01401*, 2018.

[63] Abeba Birhane, Vinay Uday Prabhu, and Emmanuel Kahembwe. Multimodal datasets: misogyny, pornography, and malignant stereotypes, 2021.

[64] Abeba Birhane and Jelle van Dijk. Robot rights? let's talk about human welfare instead. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 207–213, New York, NY, USA, 2020. Association for Computing Machinery.

[65] Ken Birman, Bharath Hariharan, and Christopher De Sa. Cloud-Hosted

Intelligence for Real-Time IoT Applications. *SIGOPS Oper. Syst. Rev.*, 53(1):7–13, July 2019.

[66] Emily Birnbaum. Advocates Urge Law Journal to Disclose Microsoft, Google Ties. *Bloomberg News*, April 2024.

[67] Emily Black, Klas Leino, and Matt Fredrikson. Selective Ensembles for Consistent Predictions. In *International Conference on Learning Representations*, 2022.

[68] Emily Black, Manish Raghavan, and Solon Barocas. Model Multiplicity: Opportunities, Concerns, and Solutions. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 850–863, New York, NY, USA, 2022. Association for Computing Machinery.

[69] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. GPT-NeoX-20B: An Open-Source Autoregressive Language Model, 2022.

[70] Patrick Blackburn, Johan F. A. K. van Benthem, and Frank Wolter. *Handbook of Modal Logic*, volume 3. Elsevier Science Inc., USA, 2006.

[71] Harvard Law Review Editorial Board. Google LLC v. Oracle America, Inc., 2021. `https://harvardlawreview.org/2021/11/google-llc-v-oracle-america-inc/`.

[72] Mark Boddy and Thomas L. Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artif. Intell.*, 67(2):245–285, June 1994.

[73] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, et al. Improving Language Models by Retrieving from Trillions of Tokens. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR, 2022.

[74] Alexei Botchkarev. A New Typology Design of Performance Metrics to Measure Errors in Machine Learning Regression Algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management*, 14:045–076, 2019.

[75] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[76] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, Jan 2018.

[77] Alexandre Bouchard-Côté, Sebastian J Vollmer, and Arnaud Doucet. The bouncy particle sampler: A nonreversible rejection-free Markov chain Monte Carlo method. *Journal of the American Statistical Association*, 113(522):855–867, 2018.

[78] Neal E. Boudette. Tesla Says Autopilot Makes Its Cars Safer. Crash Victims Say It Kills. *The New York Times*, 2021.

[79] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine Unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159, 2021.

[80] Xavier Bouthillier, César Laurent, and Pascal Vincent. Unreproducible Research is Reproducible. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 725–734. PMLR, June 2019.

[81] Mark Bovens. Analysing and assessing accountability: A conceptual framework. *European LawJjournal*, 13(4):447–468, 2007.

[82] Mark Bovens, Thomas Schillemans, and Robert E Goodin. Public accountability. *The Oxford Handbook of Public Accountability*, 1(1):1–22, 2014.

[83] Karen L Boyd. Datasheets for datasets help ml engineers notice and understand ethical issues in training data. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW2):1–27, 2021.

[84] Leo Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, August 1996.

[85] Leo Breiman. Arcing classifiers. *Annals of Statistics*, 26:801–823, 1998.

[86] Leo Breiman. Statistical Modeling: The Two Cultures. *Statistical Science*, 16(3):199–215, 2001.

[87] Kiel Brennan-Marquez. "Plausible Cuase": Explanatory Standards in the Age of Powerful Machines. *Vanderbilt Law Review*, 70:1249–1301, 2019.

[88] Eric Brewer. CAP Twelve Years Later: How the "Rules" Have Changed. *Computer*, 45(2):23–29, 2012.

[89] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC Press, 2011.

[90] Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. What Does it Mean for a Language Model to Preserve Privacy? In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 2280–2292, New York, NY, USA, 2022. Association for Computing Machinery.

[91] Nina Brown. Bots Behaving Badly: A Products Liability Approach to Chatbot-Generated Defamation. *Journal of Free Speech Law*, 3:389, 2023.

[92] Tom B. Brown, Benjamin Mann, Nick Ryder, et al. Language Models are Few-Shot Learners, 2020.

[93] Joanna J Bryson, Mihailis E Diamantis, and Thomas D Grant. Of, for, and by the people: the legal lacuna of synthetic persons. *Artificial Intelligence and Law*, 25(3):273–291, 2017.

[94] Joy Buolamwini and Timnit Gebru. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In Sorelle A. Friedler and Christo Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91, New York, NY, USA, 23–24 Feb 2018. ACM.

[95] Davide Caffagni, Manuele Barraco, Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. SynthCap: Augmenting Transformers with Synthetic Data for Image Captioning. In *International Conference on Image Analysis and Processing*, pages 112–123. Springer, 2023.

[96] Roberto Calandra, Ignasi Clavera Gilaberte, Frank Hutter, Joaquin Van-

schoren, and Jane Wang. Meta-learning. Meta-Learning Workshop at NeurIPS 2020, 2020.

[97] Ryan Calo. Robotics and the Lessons of Cyberlaw. *California Law Review*, 103(3):513–563, 2015.

[98] Ryan Calo. Modeling Through. *Duke Law Journal*, 72, 2021. SSRN Preprint.

[99] Campbell v. Acuff-Rose Music, 1994. 510 U.S. 569 (1994).

[100] Jimena Canales. *Bedeviled: A Shadow History of Demons in Science*. Princeton University Press, Princeton, NJ, USA, 2020.

[101] Capitol Records, Inc. v. MP3tunes, LLC, 2014. 48 F.Supp.3d 703 (S.D.N.Y.2014).

[102] Michael Carbin. Overparameterization: A connection between software 1.0 and software 2.0. In *3rd Summit on Advances in Programming Languages (SNAPL 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[103] Cariou v. Prince, 2013. 714 F.3d 694 (2d Cir. 2013).

[104] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting Training Data from Diffusion Models, 2023.

[105] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. Quantifying Memorization Across Neural Language Models. In *International Conference on Learning Representations*, 2023.

[106] Nicholas Carlini, Matthew Jagielski, Christopher A. Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt

Thomas, and Florian Tramèr. Poisoning Web-Scale Training Datasets is Practical, 2023.

[107] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, 2019.

[108] Nicholas Carlini, Daniel Paleka, Krishnamurthy Dj Dvijotham, Thomas Steinke, Jonathan Hayase, A. Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, Eric Wallace, David Rolnick, and Florian Tramèr. Stealing Part of a Production Language Model. *arXiv preprint arXiv:2403.06634*, 2024.

[109] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting Training Data from Large Language Models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association, August 2021.

[110] Zachariah Carmichael, Hamed F. Langroudi, Char Khazanov, Jeffrey Lillie, John L. Gustafson, and Dhireesha Kudithipudi. Performance-Efficiency Trade-off of Low-Precision Numerical Formats in Deep Neural Networks. In *Proceedings of the Conference for Next Generation Arithmetic 2019*, CoNGA'19, New York, NY, USA, 2019. Association for Computing Machinery.

[111] Cartoon Network LP, LLLP v. CSC Holdings, Inc., 2008. 536 F.3d 121 (2d Cir. 2008).

[112] Anthony Casey and Anthony Niblett. The Death of Rules and Standards, 2015. Coase-Sandor Working Paper Series in Law and Economics.

[113] Stephen Casper, Zifan Guo, Shreya Mogulothu, Zachary Marinov, Chinmay Deshpande, Rui-Jie Yew, Zheng Dai, and Dylan Hadfield-Menell. Measuring the Success of Diffusion Models at Imitating Human Artists, 2023.

[114] David M Ceperley. Path integrals in the theory of condensed helium. *Reviews of Modern Physics*, 67(2):279, 1995.

[115] Jaeyoung Cha, Jaewook Lee, and Chulhee Yun. Tighter Lower Bounds for Shuffling SGD: Random Permutations and Beyond. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[116] Kent K. Chang, Mackenzie Cramer, Sandeep Soni, and David Bamman. Speak, Memory: An Archaeology of Books Known to ChatGPT/GPT-4, 2023. `https://arxiv.org/abs/2305.00118`.

[117] 2024.

[118] Brian F. Chellas. *Modal Logic - An Introduction*. Cambridge University Press, 1980.

[119] Irene Chen, Fredrik D Johansson, and David Sontag. Why Is My Classifier Discriminatory? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[120] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li.

PixArt-α: Fast Training of Diffusion Transformer for Photorealistic Text-to-Image Synthesis, 2023.

[121] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning, 2022.

[122] Dami Choi, Christopher J. Shallue, Zachary Nado, Jaehoon Lee, Chris J. Maddison, and George E. Dahl. On Empirical Comparisons of Optimizers for Deep Learning, 2019.

[123] Jonathan H. Choi, Kristen E. Hickman, Amy Monahan, and Daniel Schwarcz. ChatGPT Goes to Law School. *Journal of Legal Education*, 2023. Forthcoming.

[124] Merlin Chowkwanyun, D. Wolfe, J. Colgrove, R. Bayer, and A. Fairchild. Beyond the Precautionary Principle: Protecting Public Health and the Environment in the Face of Uncertainty. In C.C. Macpherson, editor, *Bioethical Insights into Values and Policy*. Springer International Publishing, Switzerland, 2016.

[125] J Andrés Christen and Colin Fox. Markov chain monte carlo using an approximation. *Journal of Computational and Graphical statistics*, 14(4):795–810, 2005.

[126] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023.

[127] Danielle K. Citron and Ryan Calo. The Automated Administrative State: A Crisis of Legitimacy, 2020. Working Paper.

[128] Danielle Keats Citron. Technological Due Process. *Washington University Law Review*, 85(6):1249–1314, 2008.

[129] Danielle Keats Citron and Frank A. Pasquale. The Scored Society: Due Process for Automated Predictions. *Washington Law Review*, 89:655–690, 2014.

[130] Danielle Keats Citron and Daniel J. Solove. Privacy Harms. *Boston University Law Review*, 102, 2022.

[131] Marc Claesen and Bart De Moor. Hyperparameter Search in Machine Learning, 2015.

[132] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, April 1986.

[133] Ignacio Cofone and Katherine J. Strandburg. Strategic Games and Algorithmic Secrecy. *McGill Law Journal*, 623, 2019.

[134] P. Colangelo, N. Nasiri, E. Nurvitadhi, A. Mishra, M. Margala, and K. Nealis. Exploration of Low Numeric Precision Deep Learning Inference Using Intel FPGAs. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 73–80, Boulder, CO, USA, 2018. IEEE.

[135] Samantha Cole. 'Life or Death:' AI-Generated Mushroom Foraging Books Are All Over Amazon. *404 Media*, August 2023.

https://www.404media.co/ai-generated-mushroom-foraging-books-amazon/.

[136] A. Feder Cooper and Ellen Abrams. Emergent Unfairness in Algorithmic Fairness-Accuracy Trade-Off Research. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '21, page 46–54, New York, NY, USA, 2021. Association for Computing Machinery.

[137] A. Feder Cooper, Maria Antoniak, Christopher De Sa, Marilyn Migiel, and David Mimno. 'Tecnologica cosa': Modeling Storyteller Personalities in Boccaccio's 'Decameron'. In *Proceedings of the 5th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 147–153, Punta Cana, Dominican Republic (online), November 2021. Association for Computational Linguistics.

[138] A. Feder Cooper, Jonathan Frankle, and Christopher De Sa. Non-Determinism and the Lawlessness of Machine Learning Code. In *Proceedings of the 2022 Symposium on Computer Science and Law*, CSLAW '22, page 1–8, New York, NY, USA, 2022. Association for Computing Machinery.

[139] A. Feder Cooper and James Grimmelmann. The Files are in the Computer: Copyright, Memorization, and Generative AI. *arXiv preprint arXiv:2404.12590*, 2024.

[140] A. Feder Cooper, Wentao Guo, Khiem Pham, Tiancheng Yuan, Charlie F. Ruan, Yucheng Lu, and Christopher De Sa. Coordinating Distributed Example Orders for Provably Accelerated Training. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[141] A. Feder Cooper, Katherine Lee, Madiha Zahrah Choksi, Solon Barocas,

Christopher De Sa, James Grimmelmann, Jon Kleinberg, Siddhartha Sen, and Baobao Zhang. Arbitrariness and Social Prediction: The Confounding Role of Variance in Fair Classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(20):22004–22012, March 2024.

[142] A. Feder Cooper, Katherine Lee, James Grimmelmann, Daphne Ippolito, Christopher Callison-Burch, Christopher A. Choquette-Choo, Niloofar Mireshghallah, Miles Brundage, David Mimno, Madiha Zahrah Choksi, Jack M. Balkin, Nicholas Carlini, Christopher De Sa, Jonathan Frankle, Deep Ganguli, Bryant Gipson, Andres Guadamuz, Swee Leng Harris, Abigail Z. Jacobs, Elizabeth Joh, Gautam Kamath, Mark Lemley, Cass Matthews, Christine McLeavey, Corynne McSherry, Milad Nasr, Paul Ohm, Adam Roberts, Tom Rubin, Pamela Samuelson, Ludwig Schubert, Kristen Vaccaro, Luis Villa, Felix Wu, and Elana Zeide. Report of the 1st Workshop on Generative AI and Law. *arXiv preprint arXiv:2311.06477*, 2023.

[143] A. Feder Cooper and Karen Levy. Fast or Accurate? Governing Conflicting Goals in Highly Autonomous Vehicles. *Colorado Technology Law Journal*, 20:249–277, 2022.

[144] A. Feder Cooper, Karen Levy, and Christopher De Sa. Accuracy-Efficiency Trade-Offs and Accountability in Distributed ML Systems. In *Equity and Access in Algorithms, Mechanisms, and Optimization*, EAAMO '21, New York, NY, USA, 2021. Association for Computing Machinery.

[145] A. Feder Cooper, Yucheng Lu, Jessica Forde, and Christopher M De Sa. Hyperparameter Optimization Is Deceiving Us, and How to Stop It. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman

Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 3081–3095. Curran Associates, Inc., 2021.

[146] A. Feder Cooper, Emanuel Moss, Benjamin Laufer, and Helen Nissenbaum. Accountability in an Algorithmic Society: Relationality, Responsibility, and Robustness in Machine Learning. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 864–876, New York, NY, USA, 2022. Association for Computing Machinery.

[147] A. Feder Cooper and Gili Vidan. Making the Unaccountable Internet: The Changing Meaning of Accounting in the Early ARPANET. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 726–742, New York, NY, USA, 2022. Association for Computing Machinery.

[148] Copyright Law of the United States, October 1976. U.S.C. 17, 103.

[149] Copyright Law of the United States, December 1990. U.S.C. 17, 102.

[150] Copyright Law of the United States, October 1992. U.S.C. 17, 107.

[151] Copyright Law of the United States, November 2002. U.S.C. 17, 106.

[152] Copyright Law of the United States, December 2010. U.S.C. 17, 101.

[153] Copyright Law of the United States, December 2010. U.S.C. 17, 512.

[154] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery*

and Data Mining, KDD '17, page 797–806, New York, NY, USA, 2017. Association for Computing Machinery.

[155] Robert Cornish, Paul Vanetti, Alexandre Bouchard-Côté, George Deligiannidis, and Arnaud Doucet. Scalable Metropolis-Hastings for exact Bayesian inference with large datasets. *International Conference on Machine Learning*, 2019.

[156] CoStar Group, Inc. v. LoopNet, Inc., 2004. 373 F.3d 544 (4th Cir. 2004).

[157] National Research Council. *Risk Assessment in the Federal Government: Managing the Process*. The National Academies Press, Washington, DC, USA, 1983.

[158] National Research Council. *Science and Judgment in Risk Assessment*. The National Academies Press, Washington, DC, USA, 1994.

[159] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training Deep Neural Networks with binary weights during propagations, 2015.

[160] Kate Crawford and Jason Schultz. Big Data and Due Process: Toward a Framework to Redress Predictive Privacy Harms. *B.C. Law Review*, 55:93–128, 2014.

[161] Kathleen Creel and Deborah Hellman. The Algorithmic Leviathan: Arbitrariness, Fairness, and Opportunity in Algorithmic Decision-Making Systems. *Canadian Journal of Philosophy*, 52(1):26–43, 2022.

[162] General Tips for Designing Prompts, 2023. `https://www.promptingguide.ai/introduction/tips`.

[163] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Advances in Neural Information Processing Systems*, 2022.

[164] DC Comics v. Towle, 2015. 802 F.3d 1012 (9th Cir. 2015).

[165] Christopher De Sa. Random Reshuffling is Not Always Better. In *Advances in Neural Information Processing Systems*, 2020.

[166] Christopher De Sa, Matthew Feldman, Christopher Ré, and Kunle Olukotun. Understanding and Optimizing Asynchronous Low-Precision Stochastic Gradient Descent. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, pages 561–574, New York, NY, USA, 2017. Association for Computing Machinery.

[167] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: Amazon's Highly Available Key-value Store. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, SOSP '07, pages 205–220, New York, NY, USA, 2007. ACM.

[168] Fernando Delgado, Solon Barocas, and Karen Levy. An Uncommon Task: Participatory Design in Legal AI. *Proceedings of the ACM on Human-Computer Interaction*, 6(CSCW1), apr 2022.

[169] Karan Desai, Gaurav Kaul, Zubin Aysola, and Justin Johnson. RedCaps: Web-curated image-text data created by the people, for the people. *arXiv preprint arXiv:2111.11431*, 2021.

[170] René Descartes. *Discourse on Method and Meditations on First Philosophy*. Hackett Publishing Company, Inc., Translator Donald A. Cress, 4th edition, 1998. Meditation One: Concerning Those Things That Can Be Called into Doubt.

[171] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[172] Nicholas Diakopoulos. Accountability, Transparency, and Algorithms. *The Oxford Handbook of Ethics of AI*, 17(4):197, 2020.

[173] Thomas G. Dietterich. Robust artificial intelligence and robust human organizations. *Frontiers of Computer Science*, 13(1):1–3, Dec 2018.

[174] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. Retiring Adult: New Datasets for Fair Machine Learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 6478–6490. Curran Associates, Inc., 2021.

[175] Lisa Cingiser DiPippo and Victor Fay Wolfe. Object-based semantic real-time concurrency control with bounded imprecision. *IEEE Transactions on Knowledge and Data Engineering*, 9(1):135–147, January 1997.

[176] Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. Show Your Work: Improved Reporting of Experimen-

tal Results. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2185–2194, Hong Kong, China, November 2019. Association for Computational Linguistics.

[177] Doe 1 v. GitHub, Inc., 2022. No. 4:22-cv-06823 (N.D. Cal. November 3, 2022).

[178] Pedro Domingos. A Unified Bias-Variance Decomposition and Its Applications. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 231–238, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[179] Pedro Domingos. A unified bias-variance decomposition. Technical report, University of Washington, 2000.

[180] Finale Doshi-Velez and Been Kim. Considerations for Evaluation and Generalization in Interpretable Machine Learning. In Hugo Jair Escalante, Sergio Escalera, Isabelle Guyon, Xavier Baró, Yağmur Güçlütürk, Umut Güçlü, and Marcel van Gerven, editors, *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pages 3–17. Springer International Publishing, 2018.

[181] 2023. `https://dreamstudio.ai/`.

[182] Raaz Dwivedi and Lester Mackey. Kernel thinning. *arXiv preprint arXiv:2105.05842*, 2021.

[183] Raaz Dwivedi and Lester Mackey. Generalized Kernel Thinning. In *Tenth International Conference on Learning Representations*, 2022.

[184] B. Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 1979.

[185] Bradley Efron and Robert Tibshirani. Improvements on Cross-Validation: The 632+ Bootstrap Method. *Journal of the American Statistical Association*, 92(438):548–560, 1997.

[186] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Number 57 in Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton, Florida, USA, 1993.

[187] Madeleine Clare Elish. Moral Crumple Zones: Cautionary Tales in Human-Robot Interaction. *Engaging Science, Technology, and Society*, 2019.

[188] Charles Elkan. The Foundations of Cost-Sensitive Learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 2001.

[189] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20:1–21, 2018.

[190] E. Allen Emerson. *Temporal and Modal Logic*, page 995–1072. MIT Press, Cambridge, MA, USA, 1991.

[191] European Parliament Committee on Legal Affairs. Report with Recommendations to the Commission on Civil Law Rules on Robotics, 2017.

[192] evelyn douek. Governing Online Speech: From 'Posts-As-Trumps' to Proportionality and Probability. *Columbia Law Review*, 121(3):759–834, 2021.

[193] evelyn douek. The Siren Call of Content Moderation Formalism. *New Technologies of Communication and the First Amendment: The Internet, Social Media and Censorship*, 2022. Forthcoming.

[194] Alessandro Fabris, Stefano Messina, Gianmaria Silvello, and Gian Antonio Susto. Algorithmic Fairness Datasets: the Story so Far. *Data Mining and Knowledge Discovery*, 36(6):2074–2152, September 2022.

[195] Frank Fagan and Saul Levmore. The Impact of Artificial Intelligence on Rules, Standards, and Judicial Discretion. *Southern California Law Review*, 93(1), 2019.

[196] Gregory Falco, Ben Shneiderman, Julia Badger, Ryan Carrier, Anton Dahbura, David Danks, Martin Eling, Alwyn Goodloe, Jerry Gupta, Christopher Hart, Marina Jirotka, Henric Johnson, Cara LaPointe, Ashley J. Llorens, Alan K. Mackworth, Carsten Maple, Sigurour Emil Pálsson, Frank Pasquale, Alan Winfield, and Zee Kin Yeong. Governing AI safety through independent audits. *Nature Machine Intelligence*, 3:566–571, 2021.

[197] Federal Motor Vehicle Safety Standards. V2V Communications, 2017. Published as 82 FR 3854, from CFR 49 part 571.

[198] Federal Trade Commission. A Call For Transparency and Accountability: A Report of the Federal Trade Commission, May 2014.

[199] Joel Feinberg. Collective Responsibility. *The Journal of Philosophy*, 65(21):674–688, 1968.

[200] Joel Feinberg. *Doing & Deserving: Essays in the Theory of Responsibility*. Princeton University Press, Princeton, NJ, USA, 1970.

[201] Joel Feinberg. Sua Culpa. In *Ethical Issues in the Use of Computers*, pages 102–120. Princeton University Press, Princeton, NJ, USA, 1985.

[202] Feist Publications v. Rural Telephone Service Company, 1991. 499 U.S. 340 (1991).

[203] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and Removing Disparate Impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 259–268, New York, NY, USA, 2015. Association for Computing Machinery.

[204] Eduardo Fermé and Sven Ove Hansson. AGM 25 Years: Twenty-Five Years of Research in Belief Change. *Journal of Philosophical Logic*, 40:295–331, April 2011.

[205] Matthias Feurer and Frank Hutter. Hyperparameter optimization. In Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors, *Automated Machine Learning: Methods, Systems, Challenges*, pages 3–33. Springer International Publishing, USA, 2019.

[206] HMDA Data Publication, 2017. Released due to the Home Mortgage Disclosure Act.

[207] Michael J. Fischer and Lenore D. Zuck. Reasoning about uncertainty in fault-tolerant distributed systems. In M. Joseph, editor, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 142–158, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[208] Benjamin Fish, Jeremy Kun, and Ádám Dániel Lelkes. A Confidence-Based Approach for Balancing Fairness and Accuracy, 2016. Preprint.

[209] Mary Flanagan and Helen Nissenbaum. *Values at Play in Digital Games*. The MIT Press, 2014.

[210] Jessica Zosa Forde, A. Feder Cooper, Kweku Kwegyir-Aggrey, Chris De

Sa, and Michael Littman. Model Selection's Disparate Impact in Real-World Deep Learning Applications. *arXiv preprint arXiv:2104.00606*, 2021.

[211] Alex A. Freitas. Comprehensible classification models: A position paper. *SIGKDD Explor. Newsl.*, 15(1):1–10, March 2014.

[212] Sorelle A. Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. On the (im)possibility of fairness, 2016.

[213] Sorelle A. Friedler, Carlos Scheidegger, Suresh Venkatasubramanian, Sonam Choudhary, Evan P. Hamilton, and Derek Roth. A Comparative Study of Fairness-Enhancing Interventions in Machine Learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, page 329–338, New York, NY, USA, 2019. Association for Computing Machinery.

[214] Batya Friedman and David G. Hendry. *Value Sensitive Design: Shaping Technology with Moral Imagination*. The MIT Press, USA, 2019.

[215] Batya Friedman and Helen Nissenbaum. Bias in Computer Systems. *ACM Trans. Inf. Syst.*, 14(3):330–347, July 1996.

[216] Jeanne C Fromer. Machines as the new Oompa-Loompas: trade secrecy, the cloud, machine learning, and automation. *NYU L. Rev.*, 94:706, 2019.

[217] Masatoshi Fukushima, Yoichi Oshima, and Masayoshi Takeda. *Dirichlet forms and symmetric Markov processes*, volume 19. Walter de Gruyter, 2010.

[218] Lon L. Fuller. *The Morality of Law*. Yale University Press, New Haven, 1965.

[219] Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, et al. DataComp: In search of the next generation of multimodal datasets, 2023.

[220] Deep Ganguli, Amanda Askell, Nicholas Schiefer, et al. The Capacity for Moral Self-Correction in Large Language Models, 2023.

[221] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB Dataset of Diverse Text for Language Modeling, 2020.

[222] Hector Garcia-Molina. Using Semantic Knowledge for Transaction Processing in a Distributed Database. *ACM Transactions on Database Systems*, 8(2), June 1983.

[223] Harold Garfinkel. *Studies in Ethnomethodology*. Polity Press, Cambridge, UK, 1984.

[224] Jon Garon. An AI's Picture Paints a Thousand Lies: Designating Responsibility for Visual Libel. *Journal of Free Speech Law*, 3:425, 2023.

[225] James Garson. Modal Logic. In *The Stanford Encyclopedia of Philosophy*. Fall 2018 Edition, Edward N. Zalta (ed.), 2018.

[226] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[227] A. Gelman and Eric Loken. The garden of forking paths: Why multiple comparisons can be a problem, even when there is no "fishing expedition"

or ”p-hacking” and the research hypothesis was posited ahead of time, 2019.

[228] Andrew Gelman and Eric Loken. The statistical crisis in science: data-dependent analysis—a “garden of forking paths”–explains why many statistically significant comparisons don’t hold up. *American Scientist*, 102(6):460–466, 2014.

[229] Stuart Geman, Elie Bienenstock, and René Doursat. Neural Networks and the Bias/Variance Dilemma. *Neural Comput.*, 4(1):1–58, January 1992.

[230] Daniel Gervais. The Derivative Right, or Why Copyright Law Protects Foxes Better than Hedgehogs. *Vanderbilt Journal of Entertainment and Technology Law*, 15:785, 2013.

[231] Daniel Gervais. AI Derivatives: The Application to the Derivative Work Right to Literary and Artistic Productions of AI Machines. *Seton Hall Law Review*, 52:1111, 2022.

[232] Edmund L. Gettier. Is Justified True Belief Knowledge? *Analysis*, 23(6):121–123, 06 1963.

[233] Getty Images (US), Inc. v. Stability AI, Inc., 2023. No. 1:23-cv-00135 (D. Del. February 3, 2023).

[234] GitHub. About GitHub Copilot for Individuals, GitHub, 2023. `https://docs.github.com/en/copilot/overview-of-github-copilot/about-github-copilot-for-individuals`.

[235] GitHub. Configuring github copilot in your environment, 2023. `https://docs.github.com/en/copilot/configuring-`

github-copilot/configuring-github-copilot-in-your-environment.

[236] Aaron Gokaslan, A. Feder Cooper, Jasmine Collins, Landan Seguin, Austin Jacobson, Mihir Patel, Jonathan Frankle, Cory Stephenson, and Volodymyr Kuleshov. CommonCanvas: An Open Diffusion Model Trained with Creative-Commons Images. *arXiv preprint arXiv:2310.16825*, 2023.

[237] Wojciech Golab, Xiaozhou Li, and Mehul A. Shah. Analyzing Consistency Properties for Fun and Profit. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '11, pages 197–206, New York, NY, USA, 2011. ACM.

[238] Jake Goldenfein, Deirdre K. Mulligan, Helen F. Nissenbaum, and Wendy Ju. Through the Handoff Lens: Competing Visions of Autonomous Futures. *Berkeley Technology Law Journal*, 35:835–910, 2020.

[239] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. Compressing Deep Convolutional Networks using Vector Quantization, 2014.

[240] I.J. Good. The Population Frequencies of Species and the Estimation of Population Parameters. *Biometrika*, 40(3-4):237–264, 12 1953.

[241] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *ICLR (Poster)*, 2015.

[242] Google. Foundation Models, August 17, 2023. https://ai.google/discover/foundation-models/.

[243] Stephen Jay Gould. *The Mismeasure of Man*. Norton, New York, 1981.

[244] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pages 1311–1320. PMLR, 2017.

[245] James Grimmelmann. Regulation by Software. *The Yale Law Journal*, 114:1719–1758, 2005. Journal Note.

[246] James Grimmelmann. There's No Such Thing as a Computer-Authored Work – And It's a Good Thing, Too. *Columbia Journal of Law and the Arts*, 39:403, 2016.

[247] Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. OLMo: Accelerating the Science of Language Models, 2024.

[248] Ulrike Grömping. South German Credit Data: Correcting a Widely Used Data Set. Technical report, Beuth University of Applied Sciences Berlin, 2019.

[249] Odd Erik Gundersen and Sigbjørn Kjensmo. State of the Art: Reproducibility in Artificial Intelligence. In *AAAI*, 2018.

[250] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep Learning with Limited Numerical Precision. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1737–1746, Lille, France, 2015. JMLR.org.

[251] Mert Gürbüzbalaban, Asu Ozdaglar, and Pablo A Parrilo. Why random reshuffling beats stochastic gradient descent. *Mathematical Programming*, 186(1):49–84, 2021.

[252] Mert Gürbüzbalaban, Asuman E. Ozdaglar, and Pablo A. Parrilo. Convergence Rate of Incremental Gradient and Incremental Newton Methods. *SIAM Journal on Optimization*, 29(4):2542–2565, 2019.

[253] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. PeerReview: Practical Accountability for Distributed Systems. *SIGOPS Oper. Syst. Rev.*, 41(6):175–188, October 2007.

[254] Martin Hairer, Andrew M Stuart, Sebastian J Vollmer, et al. Spectral gaps for a Metropolis–Hastings algorithm in infinite dimensions. *The Annals of Applied Probability*, 24(6):2455–2490, 2014.

[255] Joseph Y. Halpern. *Reasoning about Uncertainty*. The MIT Press, 2 edition, 2017.

[256] Joseph Y. Halpern and Y. Shoham. A propositional modal logic of time intervals. In *Journal of the ACM*, 1991.

[257] Ronan Hamon, Henrik Junklewitz, Gianclaudio Malgieri, Paul De Hert, Laurent Beslay, and Ignacio Sanchez. Impossible Explanations? Beyond

explainable AI in the GDPR from a COVID-19 use case scenario. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 549–559, New York, NY, USA, 2021. ACM.

[258] Song Han, Huizi Mao, and William J. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *ICLR 2016*. International Conference on Learning Representations, 2016.

[259] Jeff Z. HaoChen and Suvrit Sra. Random Shuffling Beats SGD after Finite Epochs. In *Proceedings of the International Conference on Machine Learning*, volume 97, pages 2624–2633, 2019.

[260] Moritz Hardt, Eric Price, Eric Price, and Nati Srebro. Equality of Opportunity in Supervised Learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, Red Hook, NY, USA, 2016. Curran Associates, Inc.

[261] Nick Harvey and Samira Samadi. Near-Optimal Herding. In *Proceedings of The 27th Conference on Learning Theory*, volume 35, pages 1165–1182, 2014.

[262] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, USA, 2 edition, 2009.

[263] W. Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. 1970.

[264] David K. Hausman. The Danger of Rigged Algorithms: Evidence from Immigration Detention Decisions. Technical report, Stanford University, 2021.

[265] Chris Hawblitzel, Jon Howell, Manos Kapritsos, Jacob R. Lorch, Bryan Parno, Michael L. Roberts, Srinath Setty, and Brian Zill. IronFleet: Proving Practical Distributed Systems Correct. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, page 1–17, New York, NY, USA, 2015. Association for Computing Machinery.

[266] Derrick Hawkins. Researchers use facial recognition tools to predict sexual orientation. LGBT groups aren't happy. *The Washington Post*, September 2017.

[267] Aviad Heifeitz and Philippe Mongin. The Modal Logic of Probability. In *Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 175–185, July 1998.

[268] Joseph M. Hellerstein and Peter Alvaro. Keeping CALM: When Distributed Consistency is Easy. *Communications of the ACM*, 63(9):72–81, August 2020.

[269] Deborah Hellman. Big Data and Compounding Injustice, 2021. Forthcoming, SSRN preprint.

[270] Peter Henderson, Tatsunori Hashimoto, and Mark Lemley. Where's the Liability in Harmful AI Speech? *Journal of Free Speech Law*, 3:589, 2023.

[271] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning that Matters. In *Thirty-Second AAAI Conference On Artificial Intelligence*, 2018.

[272] Peter Henderson, Xuechen Li, Dan Jurafsky, Tatsunori Hashimoto, Mark A. Lemley, and Percy Liang. Foundation Models and Fair Use, 2023.

[273] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021.

[274] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[275] Laura Heymann. Everything is Transformative: Fair Use and Reader Response. *Columbia Journal of Law and the Arts*, 31:445, 2008.

[276] Kenneth Einar Himma. Artificial agency, consciousness, and the criteria for moral agency: What properties must an artificial agent have to be a moral agent? *Ethics and Information Technology*, 11(1):19–29, 2009.

[277] Jaakko Hintikka. *Knowledge and Belief*. Cornell University Press, 1962.

[278] Geoffrey E. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade: Second Edition*, pages 599–619. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[279] Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B. Gibbons, Garth A Gibson, Greg Ganger, and Eric P Xing. More Effective Distributed ML via a Stale Synchronous Parallel Parameter Server. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1223–1231. Curran Associates, Inc., USA, 2013.

[280] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[281] Eric J. Horvitz. Reasoning about Beliefs and Actions under Computational Resource Constraints. In *Proceedings of the Third Conference on Uncertainty in Artificial Intelligence*, UAI '87, pages 429–447, Seattle, WA, 1987. AUAI Press.

[282] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, 2017. arXiv preprint.

[283] Chihwei Hsu, Chihchung Chang, and ChihJen Lin. A Practical Guide to Support Vector Classification, November 2003.

[284] Wei Hu, Zhiyuan Li, and Dingli Yu. Understanding Generalization of Deep Neural Networks Trained with Noisy Labels. In *ICLR*, 2020.

[285] Kun Huang, Xiao Li, Andre Milzarek, Shi Pu, and Junwen Qiu. Distributed Random Reshuffling over Networks. *arXiv preprint arXiv:2112.15287*, 2021.

[286] HuggingFace. Models, 2023. `https://huggingface.co/models`.

[287] Ben Hutchinson, Andrew Smart, Alex Hanna, Emily Denton, Christina Greer, Oddur Kjartansson, Parker Barnes, and Margaret Mitchell. Towards accountability for machine learning datasets: Practices from software engineering and infrastructure. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 560–575, New York, NY, USA, 2021. ACM.

[288] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2018. In press, available at `http://automl.org/book`.

[289] Frank Hutter, Joaquin Vanschoren, Marius Lindauer, Charles Weill, Katharina Eggensperger, and Matthias Feurer. Icml workshop on automated machine learning. AutoML Workshop at ICML 2020, 2020.

[290] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. OpenCLIP, July 2021.

[291] Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*, 2016.

[292] Facebook Incubator. Collective communications library with various primitives for multi-machine training, 2023. `https://github.com/facebookincubator/gloo`.

[293] Technology Innovation Institute. Falcon, 2023. `https://falconllm.tii.ae/falcon.html`.

[294] Daphne Ippolito, Florian Tramèr, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher A. Choquette-Choo, and Nicholas Carlini. Preventing Verbatim Memorization in Language Models Gives a False Sense of Privacy, 2023.

[295] Charles Isbell. You Can't Escape Hyperparameters and Latent Vari-

ables: Machine Learning as a Software Engineering Enterprise. NeurIPS Keynote, 2020.

[296] Shin Ishii, Wako Yoshida, and Junichiro Yoshimoto. Control of exploitation–exploration meta-parameter in reinforcement learning. *Neural Networks*, 15(4):665–687, 2002.

[297] Matthias Jarke. Requirements Tracing. *Communications of the ACM*, 41(12):32–36, December 1998.

[298] Sheila Jasanoff. *The Ethics of Invention: Technology and the Human Future*. New York: W.W. Norton & Company, USA, 2016.

[299] Muhammad Atif Javed and Uwe Zdun. A systematic literature review of traceability approaches between software architecture and source code. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*, pages 1–10, London, England, United Kingdom, 2014. ACM Press.

[300] Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 2137–2143. PMLR, 09–12 Jul 2020.

[301] J.L. v. Alphabet Inc., 2023. No. 3:23-cv-03440-LB (N.D. Cal July 11, 2023).

[302] George H. John. Cross-Validated C4.5: Using Error Estimation for Automatic Parameter Selection. Technical report, Stanford University, Stanford, CA, USA, 1994.

[303] Severin Kacianka and Alexander Pretschner. Designing Accountable Systems. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 424–437, New York, NY, USA, 2021. Association for Computing Machinery.

[304] Kadrey v. Meta Platforms, Inc., 2023. No. 3:23-cv-03417 (N.D. Cal. July 7, 2023).

[305] Stefan Kahl, Connor M. Wood, Maximilian Eibl, and Holger Klinck. BirdNET: A Deep Learning Solution for Avian Diversity Monitoring. *Ecological Informatics*, 61:101236, 2021.

[306] Daniel Kahneman, Paul Slovic, and Amos Tversky. *Judgment under Uncertainty: Heuristics and Biases*. Cambridge University Press, New York, NY, USA, 1982.

[307] Nathan Kallus and Angela Zhou. Residual Unfairness in Fair Machine Learning from Prejudiced Data, 2018.

[308] Margot E. Kaminski. Binary Governance: Lessons from the GDPR's Approach to Algorithmic Accountability. *S. Cal. L. Rev.*, 92, 2019.

[309] Sunny Seon Kang. Algorithmic accountability in public administration: The GDPR paradox. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 32–32, New York, NY, USA, 2020. ACM.

[310] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models, 2020.

[311] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[312] Falaah Arif Khan, Denys Herasymuk, and Julia Stoyanovich. On Fairness and Stability: Is Estimator Variance a Friend or a Foe?, 2023.

[313] Been Kim and Finale Doshi-Velez. Machine Learning Techniques for Accountability. *AI Magazine*, 42(1):47–52, April 2021.

[314] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, 12 2014.

[315] Dirk P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

[316] Alexandra Kleeman. Cooking with Chef Watson, I.B.M.'s Artificial-Intelligence App. *The New Yorker*, November 2016.

[317] Jon Kleinberg, Himabindu Lakkaraju, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. Human Decisions and Machine Predictions. *The Quarterly Journal of Economics*, 133(1):237–293, 2018.

[318] Jon M. Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent Trade-Offs in the Fair Determination of Risk Scores. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 43:1–43:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[319] Kate Knibbs. The Battle Over Books3 Could Change AI Forever. *Wired*, September 2023.

[320] Wei-Yin Ko, Daniel D'souza, Karina Nguyen, Randall Balestriero, and Sara Hooker. FAIR-Ensemble: When Fairness Naturally Emerges From Deep Ensembling, 2023.

[321] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A Benchmark of in-the-Wild Distribution Shifts. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5637–5664. PMLR, 18–24 Jul 2021.

[322] Ron Kohavi. Scaling up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 202–207. AAAI Press, 1996.

[323] Nitin Kohli, Renata Barreto, and Joshua A Kroll. Translation tutorial: A shared lexicon for research and practice in human-centered software systems. In *1st Conference on Fairness, Accountability, and Transparency*, volume 7, page 7, New York, NY, USA, 2018. ACM. Tutorial.

[324] Adam J. Kolber. Smooth and Bumpy Laws. *California Law Review*, 102:655–690, 2014.

[325] Eun Bae Kong and Thomas G. Dietterich. Error-Correcting Output Coding Corrects Bias and Variance. In Armand Prieditis and Stuart Russell, editors, *Machine Learning, Proceedings of the Twelfth International Conference*

on Machine Learning, Tahoe City, California, USA, July 9-12, 1995, pages 313–321, 1995.

[326] Barteld P. Koou. Probabilistic Dynamic Epistemic Logic. *Journal of Logic Language and Information*, 12(4):381–408, 2003.

[327] Anoop Korattikara, Yutian Chen, and Max Welling. Austerity in MCMC land: Cutting the Metropolis-Hastings budget. In *International Conference on Machine Learning*, pages 181–189, 2014.

[328] Jack Kosaian, K. V. Rashmi, and Shivaram Venkataraman. Parity Models: Erasure-Coded Resilience for Prediction Serving Systems. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, pages 30–46, New York, NY, USA, 2019. Association for Computing Machinery.

[329] Jeff Kosseff. A User's Guide to Section 230, and a Legislator's Guide to Amending It (or Not). *Berkeley Technology Law Journal*, 37, 2022.

[330] Dexter Kozen. Semantics of probabilistic programs. *Journal of Computer and System Sciences*, 22(3):328–350, 1981.

[331] Dexter Kozen. A Probabilistic PDL. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, page 291–297, New York, NY, USA, 1983. Association for Computing Machinery.

[332] Sarah Kreps, R. Miles McCain, and Miles Brundage. All the News That's Fit to Fabricate: AI-Generated Text as a Tool of Media Misinformation. *Journal of Experimental Political Science*, page 1–14, 2020.

[333] Joshua A. Kroll. Outlining Traceability: A Principle for Operationalizing Accountability in Computing Systems. In *Proceedings of the 2021 ACM*

Conference on Fairness, Accountability, and Transparency, FAccT '21, page 758–771, New York, NY, USA, 2021. Association for Computing Machinery.

[334] Joshua A. Kroll, Joanna Huey, Solon Barocas, Edward W. Felten, Joel R. Reidenberg, David G. Robinson, and Harlan Yu. Accountable Algorithms. *University of Pennsylvania Law Review*, 165:633–705, 2017.

[335] Nupur Kumari, Bingliang Zhang, Sheng-Yu Wang, Eli Shechtman, Richard Zhang, and Jun-Yan Zhu. Ablating Concepts in Text-to-Image Diffusion Models, 2023.

[336] Kweku Kwegyir-Aggrey, Jessica Dai, A. Feder Cooper, John Dickerson, and Keegan Hines. Repairing Regressors for Fair Classification at Any Decision Threshold, 2023.

[337] Tuomas Kynkäänniemi, Tero Karras, Miika Aittala, Timo Aila, and Jaakko Lehtinen. The Role of ImageNet Classes in Fréchet Inception Distance. *arXiv preprint arXiv:2203.06026*, 2022.

[338] LAION-2Ben, 2022. Accessed September 23, 2023.

[339] Viktor Lakic, Luca Rossetto, and Abraham Bernstein. Link-Rot In Web-Sourced Multimedia Datasets. In *MultiMedia Modeling: 29th International Conference, MMM 2023, Bergen, Norway, January 9–12, 2023, Proceedings, Part I*, page 476–488, Berlin, Heidelberg, 2023. Springer-Verlag.

[340] Sarah Lambdan. When Westlaw Fuels ICE Surveillance: Legal Ethics in the Era of Big Data Policing. *N.Y.U. Review of Law and Social Change*, 43:255–293, 2019.

[341] Leslie Lamport. ”sometime” is Sometimes ”Not Never”: On the Temporal Logic of Programs. In *Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '80, page 174–185, New York, NY, USA, 1980. Association for Computing Machinery.

[342] B. W. Lampson. Computer security in the real world. *Computer*, 37(6):37–46, June 2004.

[343] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An Empirical Evaluation of Deep Architectures on Problems with Many factors of Variation. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 473–480, New York, NY, USA, 2007. Association for Computing Machinery.

[344] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How We Analyzed the COMPAS Recidivism Algorithm. Technical report, ProPublica, May 2016.

[345] Alan Latman. "Probative Similarity" as Proof of Copying: Toward Dispelling Some Myths in Copyright Infringement. *Columbia Law Review*, 90:1187, 1990.

[346] Benjamin Laufer, Sameer Jain, A. Feder Cooper, Jon Kleinberg, and Hoda Heidari. Four Years of FAccT: A Reflexive, Mixed-Methods Analysis of Research Contributions, Shortcomings, and Future Prospects. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 401–426, New York, NY, USA, 2022. Association for Computing Machinery.

[347] Hugo Laurençon, Lucile Saulnier, Léo Tronchon, Stas Bekman, Aman-

preet Singh, Anton Lozhkov, Thomas Wang, Siddharth Karamcheti, Alexander M. Rush, Douwe Kiela, Matthieu Cord, and Victor Sanh. OBELICS: An Open Web-Scale Filtered Dataset of Interleaved Image-Text Documents, 2023.

[348] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient BackProp. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, page 9–50, Berlin, Heidelberg, 1998. Springer-Verlag.

[349] Katherine Lee, A. Feder Cooper, and James Grimmelmann. Talkin' 'Bout AI Generation: Copyright and the Generative-AI Supply Chain. *arXiv preprint arXiv:2309.08133*, 2023.

[350] Katherine Lee, A. Feder Cooper, and James Grimmelmann. Talkin' 'Bout AI Generation: Copyright and the Generative-AI Supply Chain (The Short Version). In *Proceedings of the Symposium on Computer Science and Law*, CSLAW '24, page 48–63, New York, NY, USA, 2024. Association for Computing Machinery.

[351] Katherine Lee, A. Feder Cooper, James Grimmelmann, and Daphne Ippolito. AI and Law: The Next Generation, 2023.

[352] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating Training Data Makes Language Models Better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 8424–8445, 2022.

[353] Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong,

Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, and Daniel Haziza. xFormers: A modular and hackable Transformer modelling library, 2022. `https://github.com/facebookresearch/xformers`.

[354] David Lehr and Paul Ohm. Playing with the Data: What Legal Scholars Should Learn About Machine Learning. *U.C. Davis Law Review*, 51:653–717, 2017.

[355] Keith Lehrer. The Gettier Problem and the Analysis of Knowledge. In George Sotiros Pappas, editor, *Justification and Knowledge: New Studies in Epistemology*, pages 65–78. Springer Netherlands, Dordrecht, 1979.

[356] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, and Stephen Wolff. A Brief History of the Internet. *SIGCOMM Comput. Commun. Rev.*, 39(5):22–31, October 2009.

[357] Mark A. Lemley. How Generative AI Turns Copyright Law on its Head, 2023.

[358] Mark A. Lemley and Bryan Casey. Remedies for Robots. *The University of Chicago Law Review*, 86(5):1311–1396, 2019.

[359] Lawrence Lessig. The Law of the Horse: What Cyberlaw Might Teach. *Harvard Law Review*, 501:501–549, 1999. Journal Commentaries.

[360] Lawrence Lessig. Law Regulating Code Regulating Law. *Loyal University Chicago Law Journal*, 35(1):1–14, 2003.

[361] Lawrence Lessig. *Code 2.0*. CreateSpace, Scotts Valley, CA, 2nd edition, 2009.

[362] John Leuner. *A Replication Study: Machine Learning Models Are Capable of Predicting Sexual Orientation From Facial Images*. PhD thesis, University of Pretoria, 2019. Masters Thesis.

[363] Pierre N. Leval. Toward a Fair Use Standard. *Harvard Law Review*, 103(5):1105, 1990.

[364] Amanda Levendowski. How Copyright Law Can Fix Artificial Intelligence's Implicit Bias Problem. *Washington Law Review*, 93(2):579, 2018.

[365] David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Society, 2017.

[366] Karen EC Levy. Intimate Surveillance. *Idaho L. Rev.*, 51:679, 2014.

[367] Karen EC Levy and David Merritt Johns. When open data is a trojan horse: The weaponization of transparency in science and governance. *Big Data & Society*, 3(1), 2016.

[368] PA W Lewis and Gerald S Shedler. Simulation of nonhomogeneous Poisson processes by thinning. *Naval research logistics quarterly*, 26(3):403–413, 1979.

[369] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[370] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. Scaling Distributed Machine Learning with the Parameter Server. In *OSDI'14*, page 583–598, USA, 2014. USENIX Association.

[371] Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Luke Zettlemoyer, Omer Levy, Jason Weston, and Mike Lewis. Self-alignment with instruction backtranslation. *arXiv preprint arXiv:2308.06259*, 2023.

[372] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous Decentralized Parallel Stochastic Gradient Descent. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3043–3052. PMLR, 10–15 Jul 2018.

[373] Douglas Gary Lichtman. Uncertainty and the Standard for Preliminary Relief, 2002. John M. Olin Program in Law and Economics Working Paper No. 166.

[374] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[375] Zhiyuan Jerry Lin, Jongbin Jung, Sharad Goel, and Jennifer Skeem. The limits of human predictions of recidivism. *Science Advances*, 6(7):eaaz0652, 2020.

[376] Zachary C. Lipton and Jacob Steinhardt. Troubling Trends in Machine Learning Scholarship. *ACM Queue*, 2018.

[377] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[378] Joseph P. Liu. Copyright Law's Theory of the Consumer. *Boston College Law Review*, 44:397, 2003.

[379] London-Sire Records, Inc. v. Doe 1, 2008. 542 F. Supp. 2d 153 (D. Mass. 2008).

[380] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[381] Charles Lovering, Rohan Jha, Tal Linzen, and Ellie Pavlick. Predicting inductive biases of pre-trained models. In *International Conference on Learning Representations*, 2021.

[382] Ryan Lowe and Jan Leike. Aligning language models to follow instructions. *OpenAI*, 2023. `https://openai.com/research/instruction-following`.

[383] Haonan Lu, Kaushik Veeraraghavan, Philippe Ajoux, Jim Hunt, Yee Jiun Song, Wendy Tobagus, Sanjeev Kumar, and Wyatt Lloyd. Existential Consistency: Measuring and Understanding Consistency at Facebook. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, pages 295–310, New York, NY, USA, 2015. ACM.

[384] Yucheng Lu, Wentao Guo, and Christopher De Sa. GraB: Finding Provably Better Data Permutations than Random Reshuffling. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[385] Yucheng Lu, Si Yi Meng, and Christopher De Sa. A General Analysis of Example-Selection for Stochastic Gradient Descent. In *International Conference on Learning Representations*, 2021.

[386] Yucheng Lu, Youngsuk Park, Lifan Chen, Yuyang Wang, Christopher De Sa, and Dean Foster. Variance Reduced Training with Stratified Sampling for Forecasting Models. In *Proceedings of the International Conference on Machine Learning*, pages 7145–7155. PMLR, 2021.

[387] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs Created Equal? A Large-Scale Study. In S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, and R Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 700–709. Curran Associates, Inc., 2018.

[388] Donald MacKenzie. *Mechanizing Proof: Computing, Risk, and Trust*. MIT Press, Cambridge, MA, USA, 2001.

[389] Paul Mackun, Joshua Comenetz, and Lindsay Spell. Around Four-Fifths of All U.S. Metro Areas Grew Between 2010 and 2020. Technical report, United States Census Bureau, August 2021.

[390] Dougal Maclaurin and Ryan Prescott Adams. Firefly Monte Carlo: Exact MCMC with subsets of data. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[391] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. M4 Competition.

[392] Grigory Malinovsky, Konstantin Mishchenko, and Peter Richtárik. Server-Side Stepsizes and Sampling Without Replacement Provably Help in Federated Optimization. *arXiv preprint arXiv:2201.11066*, 2022.

[393] Susan Box Mann. The Telephone Game, 2019. https://icebreakerideas.com/telephone-game/. Accessed September 27, 2023.

[394] James Manyika. An overview of Bard: an early experiment with generative AI, August 17, 2023. https://ai.google/static/documents/google-about-bard.pdf.

[395] Charles Marx, Flavio Calmon, and Berk Ustun. Predictive Multiplicity in Classification. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6765–6774. PMLR, 13–18 Jul 2020.

[396] Mata v. Avianca, 2023. No. 22-cv-1461 (S.D.N.Y. June 22, 2023).

[397] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

[398] Peter Mattson, Vijay Janapa Reddi, Christine Cheng, Cody Coleman, Greg Diamos, David Kanter, Paulius Micikevicius, David Patterson, Guenther Schmuelling, Hanlin Tang, Gu-Yeon Wei, and Carole-Jean Wu. MLPerf: An Industry Standard Benchmark Suite for Machine Learning Performance. *IEEE Micro*, 40(2):8–16, 2020.

[399] Daniel McDuff, Tim Korjakow, Scott Cambo, Jesse Josua Benjamin, Jenny Lee, Yacine Jernite, Carlos Muñoz Ferrandis, Aaron Gokaslan, Alek Tarkowski, Joseph Lindley, A. Feder Cooper, and Danish Contractor. On

the standardization of behavioral use clauses and their adoption for responsible licensing of ai. *arXiv preprint arXiv:2402.05979*, 2024.

[400] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[401] Angelina McMillan-Major, Salomey Osei, Juan Diego Rodriguez, Pawan Sasanka Ammanamanchi, Sebastian Gehrmann, and Yacine Jernite. Reusable Templates and Guides For Documenting Datasets and Models for Natural Language Processing and Generation: A Case Study of the HuggingFace and GEM Data and Model Cards, 2021. ArXiv preprint.

[402] Alexander Meinke and Matthias Hein. Towards neural networks that provably know when they don't know, 2019. ArXiv preprint.

[403] Gábor Melis, Chris Dyer, and Phil Blunsom. On the State of the At of Evaluation in Neural Language models. In *International Conference on Learning Representations*, 2018.

[404] Peter S. Menell. In Search of Copyright's Lost Ark: Interpreting the Right to Distribute in the Internet Age. *Journal of the Copyright Society of the USA*, 59, 2011.

[405] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and Editing Factual Associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[406] Aditya Krishna Menon and Robert C Williamson. The Cost of Fairness in Binary Classification. In Sorelle A. Friedler and Christo Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 107–118, New York, NY, USA, 23–24 Feb 2018. PMLR.

[407] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. In *International Conference on Learning Representations*, 2018.

[408] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[409] Jacob Metcalf, Emanuel Moss, Ranjit Singh, Emnet Tafese, and Elizabeth Anne Watkins. A relationship and not a thing: A relational approach to algorithmic accountability and assessment documentation, 2022.

[410] Jacob Metcalf, Emanuel Moss, Elizabeth Anne Watkins, Ranjit Singh, and Madeleine Clare Elish. Algorithmic impact assessments and accountability: The co-construction of impacts. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 735–746, New York, NY, USA, 2021. ACM.

[411] Metro-Goldwyn-Mayer Studios Inc. v. Grokster, Ltd., 2005. 545 U.S. 913 (2005).

[412] Metro-Goldwyn-Mayer v. American Honda Motor Co., 1995. 900 F.Supp. 1287 (C.D. Cal. 1995).

[413] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast

computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[414] Sewon Min, Suchin Gururangan, Eric Wallace, Hannaneh Hajishirzi, Noah A. Smith, and Luke Zettlemoyer. SILO Language Models: Isolating Legal Risk In a Nonparametric Datastore, 2023.

[415] Konstantin Mishchenko, Ahmed Khaled, and Peter Richtárik. Random Reshuffling: Simple Analysis with Vast Improvements. In *Advances in Neural Information Processing Systems*, 2020.

[416] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model Cards for Model Reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 220–229, New York, NY, USA, 2019. ACM.

[417] Tom M. Mitchell. The Need for Biases in Learning Generalizations. Technical report, Rutgers University, New Brunswick, NJ, 1980. `http://www-cgi.cs.cmu.edu/~tom/pubs/NeedForBias_1980.pdf`.

[418] Sparsh Mittal. A Survey of Techniques for Approximate Computing. *ACM Computing Surveys*, 48(4), March 2016.

[419] Amirkeivan Mohtashami, Sebastian Stich, and Martin Jaggi. Characterizing & Finding Good Data Orderings for Fast Convergence of Sequential Gradient Methods. *arXiv preprint arXiv:2202.01838*, 2022.

[420] Pegah Moradi and Karen Levy. The Future of Work in the Age of AI: Displacement or Risk-Shifting? *Oxford Handbook of Ethics of AI*, pages 271–287, 2020.

[421] Thierry Moreau, Joshua San Miguel, Mark Wyse, James Bornholt, Armin Alaghi, Luis Ceze, Natalie Enright Jerger, and Adrian Sampson. A Taxonomy of General Purpose Approximate Computing Techniques. *IEEE Embed. Syst. Lett.*, 10(1):2–5, March 2018.

[422] The Mosaic ML Team. composer, 2021. `https://github.com/mosaicml/composer/`.

[423] The Mosaic ML Team. streaming, 2022. `https://github.com/mosaicml/streaming/`.

[424] Emanuel Moss, Elizabeth Anne Watkins, Ranjit Singh, Madeleine Clare Elish, and Jacob Metcalf. Assembling Accountability: Algorithmic Impact Assessment for the Public Interest, 2021. SSRN preprint.

[425] Deirdre K. Mulligan and Kenneth A. Bamberger. Saving governance-by-design. *California Law Review*, 106:697–784, June 2018.

[426] Deirdre K. Mulligan and Kenneth A. Bamberger. Procurement As Policy: Administrative Process for Machine Learning. *Berkeley Technology Law Journal*, 34:771–858, October 2019.

[427] Heather Murphy. Why Stanford Researchers Tried to Create a 'Gaydar' Machine. *The New York Times*, October 2017. `https://www.nytimes.com/2017/10/09/science/stanford-sexual-orientation-study.html`.

[428] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[429] Michael D. Murray. Generative AI Art: Copyright Infringement and Fair Use, 2023.

[430] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A Metric Learning Reality Check, 2020.

[431] Todd Mytkowicz, Amer Diwan, Matthias Hauswirth, and Peter F. Sweeney. Producing Wrong Data without Doing Anything Obviously Wrong! In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIV, page 265–276, New York, NY, USA, 2009. Association for Computing Machinery.

[432] Moin Nadeem, Anna Bethke, and Siva Reddy. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5356–5371, Online, August 2021. Association for Computational Linguistics.

[433] Tejas N. Narechania. Convergence and a Case for Broadband Rate Regulation. *Berkeley Technology Law Journal*, 37:339, 2022.

[434] Naruto v. Slater, 2018. 888 F.3d 418 (9th Cir. 2018).

[435] Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable Extraction of Training Data from (Production) Language Models. *arXiv preprint arXiv:2311.17035*, 2023.

[436] National Highway Safety Administration. Human Factors Evaluation of Level 2 and Level 3 Automated Driving Concepts, August 2015.

[437] National Transportation Safety Board. Collision Between Vehicle Controlled by Developmental Automated Driving System and Pedestrian. Technical report, US Government, December 2019. Tempe, Arizona, USA. https://www.ntsb.gov/investigations/AccidentReports/Reports/HAR1903.pdf.

[438] Deanna Needell, Rachel Ward, and Nathan Srebro. Stochastic Gradient Descent, Weighted Sampling, and the Randomized Kaczmarz algorithm. In *Advances in Neural Information Processing Systems*, pages 1017–1025, 2014.

[439] Doug Newcomb. Human Factors Evaluation of Level 2 and Level 3 Automated Driving Concepts. *Wired*, September 2021.

[440] Lily Hay Newman and Andy Greenberg. Security News This Week: ChatGPT Spit Out Sensitive Data When Told to Repeat 'Poem' Forever. *Wired*, December 2023.

[441] Meta News. Introducing Code Llama, an AI Tool for Coding, August 2023. https://about.fb.com/news/2023/08/code-llama-ai-for-coding/.

[442] Behnam Neyshabur, Srinadh Bhojanapalli, David Mcallester, and Nati Srebro. Exploring Generalization in Deep Learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, Red Hook, NY, USA, 2017. Curran Associates, Inc.

[443] Thao Nguyen, Samir Yitzhak Gadre, Gabriel Ilharco, Sewoong Oh, and

Ludwig Schmidt. Improving Multimodal Datasets with Image Captioning. *arXiv preprint arXiv:2307.10350*, 2023.

[444] Nichols v. Universal Pictures Corporation, 1930. 45 F.2d 119 (2d Cir. 1930).

[445] Helen Nissenbaum. Accountability in a Computerized Society. *Science and Engineering Ethics*, 2:25–42, 1996.

[446] Feng Niu, Benjamin Recht, Christopher Re, and Stephen J. Wright. HOGWILD!: A Lock-free Approach to Parallelizing Stochastic Gradient Descent. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 693–701, USA, 2011. Curran Associates Inc.

[447] NVIDIA. NVIDIA Collective Communication Library, 2023.

[448] Paul Ohm and Jonathan Frankle. Desirable Inefficiency. *Florida Law Review*, 70:777–836, 2019.

[449] Ngozi Okidebe. Discredited Data, 2022. Forthcoming, Cornell Law Review, Vol. 107, shared privately with the authors.

[450] OpenAI. ChatGPT: Optimizing Language Models for Dialogue, 2022. `https://openai.com/blog/chatgpt`.

[451] OpenAI. ChatGPT plugins, March 2023. `https://openai.com/blog/chatgpt-plugins`.

[452] OpenAI. Custom instructions for ChatGPT, 2023. `https://openai.com/blog/custom-instructions-for-chatgpt`.

[453] OpenAI. GPT-4 System Card, March 2023. `https://cdn.openai.com/papers/gpt-4-system-card.pdf`.

[454] OpenAI. Our approach to AI safety, April 2023. https://openai.com/blog/our-approach-to-ai-safety.

[455] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.

[456] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty under Dataset Shift. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.

[457] Susan Owicki and Leslie Lamport. Proving Liveness Properties of Concurrent Programs. *ACM Trans. Program. Lang. Syst.*, 4(3):455–495, July 1982.

[458] Owner-Operator Independent Drivers Association. Re: Docket # DOT-NHTSA-2020-0106, Framework for Automated Driving System Safety, 2020. Letter to Dr. Steven Cliff, Acting Administrator, National Highway Traffic Safety Administration.

[459] Xinghao Pan, Maximilian Lam, Stephen Tu, Dimitris Papailiopoulos, Ce Zhang, Michael I Jordan, Kannan Ramchandran, and Christopher Ré.

Cyclades: Conflict-free Asynchronous Machine Learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2568–2576. Curran Associates, Inc., USA, 2016.

[460] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings. In *1st IEEE European Symposium on Security & Privacy*. IEEE, 2016.

[461] Tara Parker-Pope. Why Is It Taking So Long to Get a Covid Vaccine for Kids? *The New York Times*, August 2021.

[462] Samir Passi and Solon Barocas. Problem Formulation and Fairness. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, page 39–48, New York, NY, USA, 2019. Association for Computing Machinery.

[463] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., Red Hook, NY, USA, 2019.

[464] Willilam F. Patry. *Patry on Copyright*. 2023.

[465] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and et al. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011.

[466] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only, 2023.

[467] Perfect 10, Inc. v. Giganews, Inc., 2017. 847 F.3d 657 (9th Cir 2017).

[468] Charles Perrow. *Normal Accidents: Living with High Risk Technologies - Updated Edition*. Princeton University Press, Princeton, New Jersey, 1999.

[469] Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 1960. 274 F.2d 487 (2d Cir. 1960).

[470] Joelle Pineau. The Machine Learning Reproducibility Checklist, March 2019. `https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf`.

[471] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On Fairness and Calibration. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[472] Amir Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, page 46–57, USA, 1977. IEEE Computer Society.

[473] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis, 2023.

[474] Jacob Portes, Alexander R Trott, Sam Havens, Daniel King, Abhinav Venigalla, Moin Nadeem, Nikhil Sardana, Daya Khudia, and Jonathan Frankle. MosaicBERT: How to Train BERT with a Lunch Money Budget. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*, 2023.

[475] Alisha Pradhan, Leah Findlater, and Amanda Lazar. "Phantom Friend" or "Just a Box with Information" Personification and Ontological Categorization of Smart Speaker-based Voice Assistants by Older Adults. In *Proceedings of the ACM on Human-Computer Interaction*, volume 3, pages 1–21, New York, NY, USA, 2019. ACM.

[476] PyTorch Contributors. DataLoader API, 2023. `https://pytorch.org/docs/stable/data.html`.

[477] Shangshu Qian, Hung Pham, Thibaud Lutellier, Zeou Hu, Jungwon Kim, Lin Tan, Yaoliang Yu, Jiahao Chen, and Sameena Shah. Are My Deep Learning Systems Fair? An Empirical Study of Fixed-Seed Training. In *Advances in Neural Information Processing Systems*, volume 34, Red Hook, NY, USA, 2021. Curran Associates, Inc.

[478] Matias Quiroz, Robert Kohn, Mattias Villani, and Minh-Ngoc Tran. Speeding up MCMC by efficient data subsampling. *Journal of the American Statistical Association*, 114(526):831–843, 2019.

[479] Matias Quiroz, Minh-Ngoc Tran, Mattias Villani, Robert Kohn, and Khue-

Dung Dang. The block-poisson estimator for optimally tuned exact subsampling mcmc. *arXiv preprint arXiv:1603.08232*, 2016.

[480] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[481] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR, 2023.

[482] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-training. Technical report, OpenAI, 2018.

[483] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

[484] Edward Raff. A Step toward Quantifying Independently Reproducible Machine Learning Research. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.

[485] Colin Raffel. Collaborative, Communal, & Continual Machine Learning, 2023. `https://colinraffel.com/talks/faculty2023collaborative.pdf`.

[486] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21:1, 2020.

[487] Inioluwa Deborah Raji, Andrew Smart, Rebecca N White, Margaret Mitchell, Timnit Gebru, Ben Hutchinson, Jamila Smith-Loud, Daniel Theron, and Parker Barnes. Closing the ai accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 33–44, New York, NY, USA, 2020. ACM.

[488] Shashank Rajput, Kangwook Lee, and Dimitris Papailiopoulos. Permutation-Based SGD: Is Random Optimal? In *International Conference on Learning Representations*, 2022.

[489] Aaditya Ramdas, Nicolas Garcia, and Marco Cuturi. On Wasserstein Two Sample Testing and Related Families of Nonparametric Tests, 2015.

[490] Benjamin Recht and Christopher Ré. Toward a Noncommutative Arithmetic-geometric Mean Inequality: Conjectures, Case-studies, and Consequences. In *Conference on Learning Theory*, volume 23, pages 11.1–11.24, 2012.

[491] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5389–5400. PMLR, 09–15 Jun 2019.

[492] Joel R. Reidenberg. Lex Informatica: The Formulation of Information Policy Rules through Technology. *Texas Law Review*, 76(3):553–594, 1997.

[493] Alex Reisner. Revealed: The Authors Whose Pirated Books are Powering Generative AI. *The Atlantic*, August 2023.

[494] Rentmeester v. Nike, Inc., 2018. 883 F.3d 1111 (9th Cir. 2018).

[495] Mark Riedl. A Very Gentle Introduction to Large Language Models without the Hype, April 2023.

[496] Carlos Riquelme, George Tucker, and Jasper Snoek. Deep Bayesian Bandits Showdown: An Empirical Comparison of Bayesian Deep Networks for Thompson Sampling. In *ICLR*, February 2018.

[497] K.T. Rodolfa, H. Lamba, and R Ghani. Empirical observation of negligible fairness–accuracy trade-offs in machine learning for public policy. *Nature Machine Intelligence*, 3, 2021.

[498] Roger Lanctot. Accelerating the Future: The Economic Impact of the Emerging Passenger Economy. Technical report, Strategy Analytics, 2017.

[499] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *2022 IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[500] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, 2015.

[501] Sudip Roy, Lucja Kot, Gabriel Bender, Bailu Ding, Hossein Hojjat, Christoph Koch, Nate Foster, and Johannes Gehrke. The Homeostasis Protocol: Avoiding Transaction Coordination Through Program Analysis. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, page 1311–1326. Association for Computing Machinery, 2015.

[502] Daniel Rudolf. Explicit error bounds for markov chain monte carlo. *arXiv preprint arXiv:1108.3201*, 2011.

[503] Christopher De Sa, Megan Leszczynski, Jian Zhang, Alana Marzoev, Christopher R. Aberger, Kunle Olukotun, and Christopher Ré. High-Accuracy Low-Precision Training. Technical report, Stanford University, 2018. `https://www.cs.cornell.edu/~cdesa/papers/arxiv2018_lpsvrg.pdf`.

[504] Christopher De Sa, Chris Re, and Kunle Olukotun. Ensuring Rapid Mixing and Low Bias for Asynchronous Gibbs Sampling. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 1567–1576, New York, New York, USA, 2016. Proceedings of Machine Learning Research.

[505] Christopher De Sa, Ce Zhang, Kunle Olukotun, and Christopher Ré. Taming the Wild: A Unified Analysis of HOGWILD!-Style Algorithms. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2674–2682, Cambridge, MA, USA, 2015. MIT Press.

[506] Noah Sachs. Rescuing the Strong Precautionary Principle from its Critics. *University of Illinois Law Review*, 2011, 2011.

[507] Abdurakhmon Sadiev, Grigory Malinovsky, Eduard Gorbunov, Igor Sokolov, Ahmed Khaled, Konstantin Burlachenko, and Peter Richtárik. Federated Optimization Algorithms with Random Reshuffling and Gradient Compression. *arXiv preprint arXiv:2206.07021*, 2022.

[508] Jathan Sadowski, Salomé Viljoen, and Meredith Whittaker. Everyone should decide how their digital data are used — Not just tech companies, 2021.

[509] SAE International. Surface Vehicle Recommended Practice. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles, 2021.

[510] Matthew Sag. Copyright Safety for Generative AI. *Houston Law Review*, 2023. Forthcoming.

[511] Adrian Sampson. *Hardware and Software for Approximate Computing*. PhD thesis, University of Washington, 2015. Ph.D. Thesis.

[512] Pamela Samuelson. Allocating Ownership Rights in Computer-Generated Works. *University of Pittsburgh Law Review*, 47:1185, 1985.

[513] Pamela Samuelson. The Quest for a Sound Conception of Copyright's Derivative Work Right. *Georgetown Law Journal*, 101:1505, 2013.

[514] Pamela Samuelson. Functionality and Expression in Computer Programs: Refining the Tests for Software Copyright Infringement. *Berkeley Technology Law Journal*, 31:1215, 2016.

[515] Pamela Samuelson. Generative AI meets copyright. *Science*, 381(6654):158–161, 2023.

[516] September 2023. https://scale.com/.

[517] Thomas Scanlon. *What We Owe to Each Other*. Belknap Press, Cambridge, MA, USA, 2000.

[518] Sarah Scheffler, Eran Tromer, and Mayank Varia. Formalizing Human Ingenuity: A Quantitative Framework for Copyright Law's Substantial Similarity. In *Proceedings of the Symposium on Computer Science and Law*, pages 37–49, 2022.

[519] Markus Schlosser. Agency. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, USA, Winter 2019 edition, 2019.

[520] Mark Schmidt, Nicolas Le Roux, and Francis R. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

[521] F. Schneider, L. Balles, and P. Hennig. DeepOBS: A Deep Learning Optimizer Benchmark Suite. In *7th International Conference on Learning Representations (ICLR)*. ICLR, May 2019.

[522] Christoph Schuhmann. LAION-400-Million Open Dataset, 2021.

[523] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5B: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[524] Charles M. Schultz. Snoopy_Peanuts, 2020. https://en.wikipedia.org/wiki/Snoopy#/media/File:Snoopy_Peanuts.png. Accessed September 26, 2023.

[525] Dana Scott. Advice on modal logic. In Karel Lambert, editor, *Philosophical Problems in Logic: Some Recent Developments*, pages 143–173. Springer Netherlands, Dordrecht, 1970.

[526] D. Sculley, Jasper Snoek, Alex Wiltschko, and Ali Rahimi. Winner's Curse? On Pace, Progress, and Empirical Rigor. In *ICLR 2018 Workshop*, February 2018.

[527] Krister Segerberg. Two Traditions in the Logic of Belief: Bringing them Together. In Hans Jürgen Ohlbach and Uwe Reyle, editors, *Logic, Language and Reasoning: Essay in Honour of Dov Gabbay*, pages 135–148. Springer Science+Business Media, Dordrecht, Netherlands, 1999.

[528] Daniel Seita, Xinlei Pan, Haoyu Chen, and John Canny. An efficient minibatch acceptance test for Metropolis-Hastings. *Uncertainty in Artificial Intelligence*, 2017.

[529] Andrew D Selbst. An institutional view of algorithmic impact assessments. *Harvard Journal of Law & Technology*, 35, 2021.

[530] Andrew D. Selbst, danah boyd, Sorelle A. Friedler, Suresh Venkatasubramanian, and Janet Vertesi. Fairness and Abstraction in Sociotechnical Systems. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, page 59–68, New York, NY, USA, 2019. Association for Computing Machinery.

[531] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86—96, 2016.

[532] Hetan Shah. Algorithmic Accountability. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2128), 2018.

[533] Zechao Shang, Jeffrey Xu Yu, and Aaron J. Elmore. RushMon: Real-time Isolation Anomalies Monitoring. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, pages 647–662, New York, NY, USA, 2018. ACM.

[534] 2023. `https://sharegpt.com/`.

[535] Catherine M Sharkey. Can Data Breach Claims Survive the Economic Loss Rule. *DePaul Law Review*, 66:339, 2016.

[536] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, 2018.

[537] Hong Shen, Wesley H Deng, Aditi Chattopadhyay, Zhiwei Steven Wu, Xu Wang, and Haiyi Zhu. Value Cards: An Educational Toolkit for Teaching Social Impacts of Machine Learning through Deliberation. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 850–861, New York, NY, USA, 2021. ACM.

[538] David Shoemaker. Attributability, answerability, and accountability: Toward a wider theory of moral responsibility. *Ethics*, 121(3):602–632, 2011.

[539] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The Curse of Recursion: Training on Generated Data Makes Models Forget, 2023.

[540] Sid & Marty Krofft Television v. McDonald's Corp., 1977. 562 F.2d 1157 (1977).

[541] Sinclair v. Ziff Davis, LLC, 2020. 454 F.Supp.3d 342 (S.D.N.Y. 2020).

[542] K Sinha, J Pineau, J Forde, R N Ke, and H Larochelle. NeurIPS 2019 Reproducibility Challenge. *ReScience*, 6(2), 2020.

[543] Prabhu Teja Sivaprasad, Florian Mai, Thijs Vogels, Martin Jaggi, and François Fleuret. Optimizer Benchmarking Needs to Account for Hyperparameter Tuning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9036–9045. PMLR, 13–18 Jul 2020.

[544] Skidmore v. Zeppelin, 2020. 952 F.3d 1051 (9th Cir. 2020).

[545] Mona Sloane, Emanuel Moss, Olaitan Awomolo, and Laura Forlano. Participation is not a Design Fix for Machine Learning, 2020. ArXiv preprint.

[546] Mona Sloane, Emanuel Moss, and Rumman Chowdhury. A Silicon Valley Love Triangle: Hiring Algorithms, Pseudo-Science, and the Quest for Auditability, 2021. ArXiv preprint.

[547] Brian Cantwell Smith. The Limits of Correctness. *SIGCAS Comput. Soc.*, 14,15(1,2,3,4):18–26, January 1985.

[548] Henry E. Smith. Equity as Second-Order Law: The Problem of Opportunism, 2015. Harvard Public Law Working Paper No. 15-13.

[549] Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don't Decay the Learning Rate, Increase the Batch Size. In *International Conference on Learning Representations*, 2018.

[550] Raymond M. Smullyan. Logicians Who Reason about Themselves. In *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning about Knowledge*, TARK '86, page 341–352, San Francisco, CA, USA, 1986. Morgan Kaufmann Publishers Inc.

[551] Benjamin L.W. Sobel. Artificial Intelligence's Fair Use Crisis. *Columbia Journal of Law and The Arts*, 41:45, 2017.

[552] Benjamin L.W. Sobel. Elements of Style: A Grand Bargain for Generative AI, 2023. On file with the authors.

[553] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathany, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[554] Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E.

Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research, 2024.

[555] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *International Journal of Computer Vision*, pages 1–40, 2022.

[556] Evangelos Spiliotis, Andreas Kouloumos, Vassilios Assimakopoulos, and Spyros Makridakis. Are forecasting competitions data representative of the reality? *International Journal of Forecasting*, 36(1):37–53, 2020.

[557] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[558] Stability AI. Stable Diffusion 2.0 Release, 2022. `https://stability.ai/blog/stable-diffusion-v2-release`.

[559] Stability AI. Stable Diffusion v2-base Model Card, 2022. `https://huggingface.co/stabilityai/stable-diffusion-2-base`.

[560] Robert Stalnaker. On Logics of Knowledge and Belief. *Philosophical Studies: An International Journal for Philosophy in the Analytic Tradition*, 128(1):169–199, 2006.

[561] Luke Stark and Jevan Hutson. Physiognomic Artificial Intelligence, 2021. SSRN preprint.

[562] Merity Stephen, Xiong Caiming, Bradbury James, and Richard Socher. Pointer sentinel mixture models. *Proceedings of ICLR*, 2017.

[563] Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. Mitigating Gender Bias in Natural Language Processing: Literature Review. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1630–1640, Florence, Italy, July 2019. Association for Computational Linguistics.

[564] Cass R. Sunstein. Hazardous Heuristics, 2002. U Chicago Law & Economics Working Paper.

[565] Cass R. Sunstein. Beyond the Precautionary Principle, 2003. U Chicago Law & Economics Working Paper.

[566] Harry Surden and Mary-Anne Williams. Technological Opacity, Predictability, and Self-Driving Cars. *Cardozo Law Review*, 38:121–181, 2016.

[567] Daniel Susser. Decision Time: Normative Dimensions of Algorithmic Speed. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 1410–1420, New York, NY, USA, 2022. Association for Computing Machinery.

[568] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR (Poster)*, 2014.

[569] Matthew Talbert. Moral Responsibility. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, USA, Winter 2019 edition, 2019.

[570] Brian Z. Tamanaha. *On the Rule of Law: History, Politics, Theory*. Cambridge University Press, Cambridge, 2004.

[571] The Vicuna Team. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. *LMSYS Org*, March 2023. https://lmsys.org/blog/2023-03-30-vicuna/.

[572] TensorBoard: TensorFlow's visualization toolkit, 2023. https://www.tensorflow.org/tensorboard.

[573] Piotr Tereszkiewicz. Digital platforms: regulation and liability in the EU law. *European Review of Private Law*, 26(6), 2018.

[574] Thaler v. Perlmutter, 2023. No. 22-1564 (D.D.C. August 18, 2023).

[575] The US Copyright Office. Works Not Protected by Copyright (Circular 33), 2021. https://www.copyright.gov/circs/circ33.pdf.

[576] The White House. Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence, October 2023.

[577] David Thiel. Identifying and eliminating csam in generative ml training data and models. Technical report, Stanford University, 2023.

[578] Richmond H. Thomason. Combinations of tense and modality. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic: Volume II: Extensions of Classical Logic*, pages 135–165. Springer Netherlands, Dordrecht, 1984.

[579] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: The

new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

[580] Together. RedPajama: An Open Source Recipe to Reproduce LLaMA training dataset, 2023.

[581] Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. LLaMA: Open and Efficient Foundation Language Models, 2023.

[582] Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023.

[583] Tremblay v. OpenAI, Inc., 2023. No. 3:23-cv-03223 (N.D. Cal. June 28, 2023).

[584] Laurence H. Tribe. Trial by Mathematics: Precision and Ritual in the Legal Process. *Harvard Law Review*, 84(6):1329–1393, 1971.

[585] J. Tsitsiklis, D. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, 1986.

[586] Sherry Turkle. *The second self: Computers and the human spirit*. MIT Press, Cambridge, MA, USA, 2005.

[587] U.S. Department of Transportation. U.S. Department of Transportation Issues Advance Notice of Proposed Rulemaking to Begin Implementation of Vehicle-to-Vehicle Communications Technology, 2014. CFR 49 part 571.

[588] U.S. Government Accountability Office. Artificial Intelligence: An Accountability Framework for Federal Agencies and Other Entities., June 2021. `https://www.gao.gov/assets/gao-21-519sp.pdf`.

[589] Reeve v. Dennett, 1887. 145 Mass. 23, 28.

[590] Johan van Benthem. Epistemic Logic and Epistemology: The State of Their Affairs. *Philosophical Studies: An International Journal for Philosophy in the Analytic Tradition*, 128(1):49–76, 2006.

[591] Johan van Benthem. Dynamic logic for belief revision. *Journal of Applied Non-Classical Logics*, 17(2):129–155, 2007.

[592] Johan van Benthem, Jelle Gerbrandy, and Barteld P. Kooi. Dynamic Update with Probabilities. *Stud Logica*, 93(1):67–96, 2009.

[593] Kees-Jan van Dorp. Tracking and tracing: a structure for development and contemporary practices. *Logistics Information Management*, 15(1):24–33, March 2002.

[594] Diane Vaughan. *The Challenger launch Decision: Risky Technology, Culture, and Deviance at NASA*. University of Chicago Press, Chicago, IL, USA, 1996.

[595] Briana Vecchione, Karen Levy, and Solon Barocas. Algorithmic auditing and social justice: Lessons from the history of audit studies. In *Equity and Access in Algorithms, Mechanisms, and Optimization*, pages 1–9. ACM, New York, NY, USA, 2021.

[596] Carissa Véliz. Moral zombies: why algorithms are not moral agents. *AI & SOCIETY*, 36, 2021.

[597] Abhinav Venigalla and Linden Li. Mosaic LLMs (Part 2): GPT-3 quality for ¡$500k. *MosaicML*, September 2022. `https://www.mosaicml.com/blog/gpt-3-quality-for-500k`.

[598] Salomé Viljoen. A relational theory of data governance. *Yale Law Journal*, 131(2), 2021.

[599] James Vincent. Getty Images is suing the creators of AI art tool Stable Diffusion for scraping its content. *The Verge*, January 2023.

[600] James Vincent. Meta's powerful AI language model has leaked online — what happens now? *The Verge*, 2023. `https://wandb.ai/site`.

[601] Lee Vinsel. *Moving Violations: Automobiles, Experts, and Regulations in the United States*. Johns Hopkins University Press, Baltimore, 2019.

[602] Werner Vogels. Eventually Consistent. *Communications of the ACM*, 52(1):40–44, January 2009.

[603] Eugene Volokh. Large Libel Models? Liability for AI Outpu. *Journal of Free Speech Law*, 3:489, 2023.

[604] Carl von Clausewitz. *Vom Kriege*. Ferdinand Dümmler, 1832.

[605] Nikhil Vyas, Sham Kakade, and Boaz Barak. On Provable Copyright Protection for Generative Models, 2023.

[606] Sandra Wachter and Brent Mittelstadt. A Right to Reasonable Inferences: Re-Thinking Data Protection Law in the Age of Big Data and AI. *Columbia Business Law Review*, 2, 2019.

[607] Sandra Wachter, Brent Mittelstadt, and Luciano Floridi. Transparent, explainable, and accountable ai for robotics. *Science (Robotics)*, 2(6), 2017.

[608] Stefan Wager. Cross-Validation, Risk Estimation, and Model Selection: Comment on a Paper by Rosset and Tibshirani. *Journal of the American Statistical Association*, 115(529):157–160, 2020.

[609] Ari Ezra Waldman. Power, Process, and Automated Decision-Making. *Fordham Law Review*, 88, 2019.

[610] Walker v. Time Life Films, Inc., 1986. 784 F.2d 44 (2d Cir. 1986).

[611] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[612] Amy B. Wang. A suspect tried to blend in with 60,000 concertgoers. China's facial-recognition cameras caught him, April 2018.

[613] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan. Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205–1221, 2019.

[614] Yilun Wang and Michal Kosinski. Deep neural networks are more accurate than humans at detecting sexual orientation from facial images, 02 2018.

[615] Gary Watson. Two Faces of Responsibility. *Philosophical Topics*, 24(2):227–248, 1996.

[616] Jamelle Watson-Daniels, David C. Parkes, and Berk Ustun. Predictive Multiplicity in Probabilistic Classification, 2022.

[617] 2023. `https://wandb.ai/site`.

[618] W. E. Weihl. Commutativity-based concurrency control for abstract data types. *IEEE Transactions on Computers*, 37(12):1488–1505, 1988.

[619] MC Weinstein, KA Freedberg, EP Hyle, and AD Paltiel. Waiting for Certainty on Covid-19 Antibody Tests - At What Cost?, 2020.

[620] Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1121–1128, 2009.

[621] Jan Whittington, Ryan Calo, Mike Simon, Jesse Woo, Meg Young, and Peter Schmiedeskamp. Push, pull, and spill: A transdisciplinary case study in municipal open government. *Berkeley Technology Law Journal*, 30(3):1899–1966, 2015.

[622] Maranke Wieringa. What to account for when accounting for algorithms: a systematic literature review on algorithmic accountability. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 1–18, New York, NY, USA, 2020. ACM.

[623] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The Marginal Value of Adaptive Gradient Methods in Machine Learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4148–4158. Curran Associates, Inc., 2017.

[624] BigScience Workshop et al. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model, 2023.

[625] Xiaolin Wu and Xi Zhang. Automated Inference on Criminality using Face Images, 2016. ArXiv preprint.

[626] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[627] Feiyang Xiao, Qiaoxi Zhu, Jian Guan, Xubo Liu, Haohe Liu, Kejia Zhang, and Wenwu Wang. Synth-AC: Enhancing Audio Captioning with Synthetic Supervision. *arXiv preprint arXiv:2309.09705*, 2023.

[628] Yuanzhong Xu, HyoukJoong Lee, Dehao Chen, Hongjun Choi, Blake Hechtman, and Shibo Wang. Automatic cross-replica sharding of weight update in data-parallel training. *arXiv preprint arXiv:2004.13336*, 2020.

[629] Yichen Yang and Martin Rinard. Correctness Verification of Neural Networks, 2019. NeurIPS 2019 Workshop on Machine Learning with Guarantees.

[630] I.C. Yeh and C.H. Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36:2473–2480, 2009.

[631] Bicheng Ying, Kun Yuan, Stefan Vlaski, and Ali H. Sayed. On the performance of random reshuffling in stochastic learning. In *2017 Information Theory and Applications Workshop (ITA)*, pages 1–5. IEEE, 2017.

[632] Cristobal Young. Model Uncertainty and the Crisis in Science. *Socius*, 4:2378023117737206, 2018.

[633] Meg Young, Luke Rodriguez, Emily Keller, Feiyang Sun, Boyang Sa, Jan Whittington, and Bill Howe. Beyond open vs. closed: Balancing individual privacy and public accountability in data sharing. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 191–200, New York, NY, USA, 2019. ACM.

[634] Haifeng Yu and Amin Vahdat. Design and Evaluation of a Continuous Consistency Model for Replicated Services. In *Proceedings of the 4th Conference on Symposium on Operating System Design & Implementation - Volume 4*, OSDI'00, Berkeley, CA, USA, 2000. USENIX Association.

[635] Haifeng Yu and Amin Vahdat. Efficient Numerical Error Bounding for Replicated Network Services. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 123–133, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[636] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3):5, 2022.

[637] Binhang Yuan, Yongjun He, Jared Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy S Liang, Christopher Re, and Ce Zhang. Decentralized training of foundation models in heterogeneous environments. *Advances in Neural Information Processing Systems*, 35:25464–25477, 2022.

[638] Chulhee Yun, Shashank Rajput, and Suvrit Sra. Minibatch vs Local SGD with Shuffling: Tight Convergence Bounds and Beyond. In *International Conference on Learning Representations*, 2021.

[639] Chulhee Yun, Suvrit Sra, and Ali Jadbabaie. Open Problem: Can Single-Shuffle SGD be Better than Reshuffling SGD and GD? In *Conference on Learning Theory*, 2021.

[640] Du Zhang and Jeffrey JP Tsai. Machine learning and software engineering. *Software Quality Journal*, 11(2):87–119, 2003.

[641] Ruqi Zhang, A. Feder Cooper, and Christopher M De Sa. Asymptotically Optimal Exact Minibatch Metropolis-Hastings. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19500–19510. Curran Associates, Inc., 2020.

[642] Ruqi Zhang, A. Feder Cooper, and Christopher De Sa. AMAGOLD: Amortized Metropolis Adjustment for Efficient Stochastic Gradient MCMC. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2142–2152. PMLR, 2020.

[643] Ruqi Zhang and Christopher M De Sa. Poisson-Minibatching for Gibbs Sampling with Convergence Rate Guarantees. In *Advances in Neural Information Processing Systems*, pages 4923–4932, 2019.

[644] Wei Zhang, Suyog Gupta, Xiangru Lian, and Ji Liu. Staleness-Aware Async-SGD for Distributed Deep Learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 2350–2356. AAAI Press, 2016.

[645] Yiming Zhang and Daphne Ippolito. Prompts Should not be Seen as Secrets: Systematically Measuring Prompt Extraction Attack Success, 2023.

[646] Ritchie Zhao, Christopher De Sa, and Zhiru Zhang. Overwrite quantization: Opportunistic outlier handling for neural network accelerators, 2019.

[647] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. *CVPR*, 2018.

[648] Jonathan Zittrain. *The Future of the Internet–And How to Stop It*. Yale University Press, USA, 2008.

[649] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.