Regulating Accuracy-Efficiency Trade-Offs in Distributed Machine Learning Systems

A. Feder Cooper
Cornell University, Department of Computer Science
afc78@cornell.edu

Karen Levy
Cornell University, Department of Information Science
& Cornell Law School
karen.levy@cornell.edu

Christopher De Sa
Cornell University, Department of Computer Science
cmd353@cornell.edu

Abstract

In this paper we discuss the trade-off between accuracy and efficiency in distributed machine learning (ML) systems and analyze its resulting policy considerations. This trade-off is in fact quite common in multiple disciplines, including law and medicine, and it applies to a wide variety of subfields within computer science. Accuracy and efficiency trade-offs have unique implications in ML algorithms because, being probabilistic in nature, such algorithms generally exhibit error tolerance. After describing how the trade-off takes shape in real-world distributed computing systems, we show the interplay between such systems and ML algorithms, explaining in detail how accuracy and efficiency interact particularly in distributed ML systems. We close by making specific calls to action for approaching regulatory policy for the emerging technology of real-time distributed ML systems.

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 in engineering, particularly in the field of computer science, is between *accuracy* and *efficiency*. More specifically, there is an inherent tension between *how correct* computations are and *how long* it takes to compute them.[1]

While this trade-off represents a general problem, it plays out in various ways across different subfields of computing. For example, in computer hardware, circuits can use approximation techniques to relax constraints on accuracy—on how they perform bitwise computations—in order to speed up performance. In image processing, varying numbers of pixels can be used to represent a given image; using fewer pixels causes a loss in accuracy of the image being represented, but also furthers space-efficiency by requiring less memory to store the image. These kinds of examples are abundant in computing. In fact, the trade-off is so ubiquitous and well-known to computer scientists that it has even given rise to its own subfield, *approximate computing*, which resides within the programming languages (PL) discipline [60, 61]. This subfield has shown that it is useful to analyze the accuracy-efficiency trade-off in the context of how *error tolerant* an application is—that is, how different domains resolve the question of how much inaccuracy can be permitted, while retaining guarantees about quality and safety[76].

While commonly acknowledged in some areas of computer science—perhaps to the point of mundanity—the policy implications of this trade-off remain relatively unexamined. We therefore focus this paper on analyzing its regulatory implications in the context of a particularly urgent technological domain—*distributed machine learning*. We argue that the accuracy-efficiency trade-off exposes a high-level abstraction that policymakers should use to provide regulatory interventions. That is, rather than operating at one of two extremes—either solely having policymakers rely on technical experts to make high-stakes policy decisions or inundating policymakers with underlying low-level technical details—we advocate for something in between: ML systems researchers should focus on providing guarantees concerning correctness and performance, and should build associated tools to help policymakers reason about these guarantees. These tools should effectively expose the degree of uncertainty in distributed ML systems, thus facilitating lawmakers' ability to reason about and regulate the resulting risk of their deployment in high-stakes domains, such as autonomous cars and unmanned military drones.

To make this case, we organize the remainder of this paper as follows. In Section 2, we outline the general trade-off between accuracy and efficiency. We discuss how this trade-off is salient in disciplines other than computing (Section 2.1) and then pay particular attention to the various ways it can be used to analyze different subfields of CS (Section 2.2). We then provide grounding for understanding how accuracy and efficiency are in tension with each other in distributed ML systems (Section 4). To understand the specifics of the trade-off in that setting, we first outline separately how it

[1] Framing the accuracy-efficiency trade-off as a cardinal trade-off in computing importantly differs from how Ohm and Frankle [67] discuss efficiency. Their work calls 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." Instead, viewing the accuracy-efficiency *trade-off* as central enables us to not refer to "inefficient" computing models as exceptional and strikes us as a more precise and generalizable statement of the issues at stake. Therefore, rather than casting particularly inefficient computing models (e.g., cryptography) as exceptional—as Ohm and Frankle do—we can conceive of them as implementing the trade-off at one end of the accuracy-efficiency spectrum (privileging accuracy).

unfolds in ML algorithms (Section 3) and distributed computing systems (Section 4.1), and then bring this discussion together to clarify emergent tensions in the subset of ML systems that serve high-stakes, real-time applications (Section 4.2). Based on the overarching themes we extract in our discussion, we then close with specific calls to action regarding these systems, both in ML systems research and in policy (Section 5).

2 The Price of Accuracy

Gathering increasingly accurate information is a computationally expensive task that is in tension with acting efficiently. In Section 2.1 we discuss how this tension plays out in various domains outside of computing, each with their own normative concerns. We then turn our attention in Section 2.2 to the various accuracy-efficiency trade-offs that are inherent throughout CS. Studying this trade-off in computing generally falls under the area of *approximate computing*. We use this discussion to ground our treatment in later sections of how such trade-offs play out uniquely in the context of machine learning (Section 3) and distributed machine learning systems (Section 4).

2.1 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 diverse range of disciplines, including economics, law, and medicine. 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 based on information is itself an action—one that can have potentially more negative consequences than acting earlier on imperfect information. Kahneman and Tversky elaborate on this idea in their well-known cognitive psychology research concerning reasoning about uncertainty [48]. They 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.

Sunstein connects this idea directly to the potential hazards of using heuristics in legal decision-making [81]. Nevertheless, he observes that heuristics are common—and necessary—in the law to obtain a suitable balance between efficient resolution and the "best" (i.e., most accurate) adjudicative outcomes.

A number of rules in US civil and criminal procedure (e.g., speedy trial requirements, local rules imposing 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. Due process implicates both values. Some areas of law explicitly consider how to balance between them. For example, the standard for preliminary injunctive relief in the United States explicitly considers 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.

Debates about the merits of the so-called "precautionary principle" in policymaking also capture the accuracy-efficiency trade-off. The precautionary principle advises extreme caution around new innovations when there is substantial unknown risk; in operation, it effectively 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, rather than vice versa. As with speedy trials, there is a trade-off between the time it takes to gather evidence—to highlight the risk landscape—and capturing that landscape effectively. 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 [82]. 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 [75].[2]

Another urgent example of the trade-off in action arises in public health—specifically, in the context of the COVID-19 pandemic and 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" [90]. More generally, the authors claim that it is the norm for healthcare practitioners to act on incomplete

[2] In addition to these examples from the law, the accuracy-efficiency trade-off is salient in other aspects of governance. In particular, it plays an important role in wartime intelligence gathering. There is an 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. This is the so-called "fog of war" notion, which attempts to capture the relationship between how the uncertainty of information changes over time [87].

information—to balance potential inaccuracies in available data with the urgency to treat serious conditions. To see this, consider an example in which a patient has a small growth on their lungs, and it is uncertain whether the growth is benign or malignant. A doctor may then be faced with the following choice: they can either perform a biopsy—a very invasive procedure—now, with the possibility that such an early stage test could yield inconclusive results; or, they can wait and see if it grows, and then biopsy it at a later stage where there can be more certain results regarding potential malignancy. The doctor is faced with a trade-off between potentially inaccurate information in the short-term and a higher certainty of accuracy (with the potential danger of not having acted quickly enough) in the longer term.

These examples all concern reasoning under uncertainty, and they tease at the relationship between uncertainty and externalities of risk. Before we discuss this relationship more formally in the context of machine learning systems (Section 4), we demonstrate how the accuracy-efficiency trade-off is widely relevant across various domains in computing.

2.2 Trading off Accuracy and Efficiency in Computing

In computing, the accuracy-efficiency trade-off is a spectrum, not a binary decision, that has implications in almost all computer science subfields (See Figure 1). To capture the intuition, let us consider a familiar example—image compression to generate JPEG images. Raw images tend to be very high resolution, meaning that they contain many pixels per inch. In order to capture each pixel, such images tend to require a lot of storage space. However, it is often not necessary to use so many pixels for a high-quality image. To the human eye, a compressed version of the raw image often suffices; removing some pixels or averaging and combining the values of neighboring pixels often is not detectable. Moreover, compressing a raw image to a JPEG takes up less computer storage space and can lead to faster image processing when doing photo editing since there are fewer pixels to consider. In other words, reducing the accuracy of the image can lead to greater computational efficiencies when manipulating it.

The notion of this trade-off between accuracy and efficiency forms the basis of the field of *approximate computing*. The main idea of this field is that a computer system can achieve certain performance benefits if it exerts less computational effort to compute perfectly accurate answers. In other words, it is possible to *relax* accuracy in order to yield efficiency improvements [60, 61, 76]. As with JPEGs, relaxing the accuracy does not necessarily have negative consequences; rather, it is possible that the decreased accuracy has no observable impact for a particular application. In other words, some applications are tolerant of inaccuracy; they are error resilient.

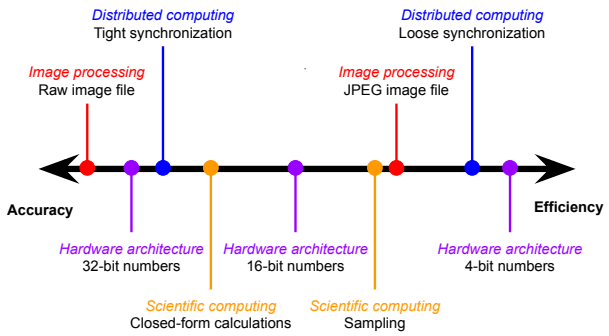

Figure 1. Examples of implementing the accuracy-efficiency trade-off in approximate computing: Image compression (Section 2.2), bit precision representing numbers (Sections 2.2 & 3.2), distributed systems (Section 4.1), and sampling (Section 3.4)

.

While the term "approximate computing" is a fairly recent innovation, these ideas are as old as computers. One of the oldest examples comes from how computer hardware represents numerical values—particularly floating-point numbers, which, as opposed to integers, can have arbitrary precision; they require (potentially infinite) decimal places to express. However, computers require discretization to store numbers in binary encoding; floating-point numbers are expressed in a finite number of bits, limiting their precision and how accurately they can reflect the values of the numbers they represent [33, 61].

Approximate computing not only demonstrates the existence of this trade-off, but also provides ways of formally characterizing it. In turn, this characterization can enable computer scientists to leverage the "right" trade-off in different application domains. For example, formal reasoning around the trade-off can yield application-specific quality metrics. Quality can be thought of as how a program conceives of "good enough" results. Often, the quality of "good enough" cannot be guaranteed with complete certainty, but can be verified with high probability. Leaving room for uncertainty allows for edge case behaviors, which might 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 therefore design software that requires a certain degree of program quality with a certain (high) probability [76].[3]

[3] A popular 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 [1].

2.3 High-Stakes Application Domains

Quality metrics are particularly salient in high-impact application areas. Consider an autonomous surgical robot or autonomous car. Both application domains require both accuracy and efficiency in order to be safe and reliable, but cannot maximize both properties at the same time due to the inherent trade-off between them. Instead, in each application domain, we need to have a sense of how much error we can tolerate in order to meet certain speed demands.

The same can be said for police use of facial recognition software. For example, as described in Dietterich [27], a study in South Wales found a false positive rate of 91.3% in a facial recognition application that tries to match faces with outstanding arrest warrants at public events.[4] While this application is not necessarily efficiency-sensitive—a human could, in theory, intervene to verify the accuracy of the results prior to acting—this would not necessarily be the case if such technology were integrated into police body cameras for the purpose of making in-the-moment (and potentially life-or-death) decisions. In situations of imminent danger, efficiency is crucial; for example, it is necessary to speedily identify a person-of-interest. Accuracy in this case is equally important; in heightened stress environments, mistaking someone for a person-of-interest has repeatedly proven catastrophic, particularly in the United States. Because of these competing technological goals, it is not clear exactly how approximate computing could be safe in this context, as the high stakes involved do not lend themselves to error resilience. In other high-impact legal contexts, the trade-off can potentially be reasoned about safely. Consider automated risk assessment tools [19, 80]. Accuracy in assessing risk is critical, but is not necessarily time-sensitive. Operating on the scale of minutes, hours, or even days might suffice, particularly if such time spans entail increases in accuracy.[5]

3 Accuracy-Efficiency Considerations in Machine Learning Algorithms

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 [12, 45]. In particular, in a classic paper, Horvitz poses the question of how autonomous agents can effectively perform computations under tight computational resource constraints [45]. He discusses how approximations or heuristics can lead to more efficient resource utilization—at the cost of potentially less-correct computation. He frames this as a "time-precision tradeoff,"[6] in order to indicate how there is an inherent tension between the utility of a correct computation and how fast that computation is completed, in the context of evaluating reasoning under uncertainty for autonomous agents.

This trade-off persists beyond classical AI to contemporary work in statistical ML, as ML's probabilistic nature has important implications for the relationship between accuracy and efficiency in ML models. Trained ML models perform inference that is not always correct, often tolerating a certain degree of inaccuracy. Being resilient to errors is necessary for producing robust models. This notion of error resilience (or inaccuracy tolerance) varies for different types of ML algorithms. Regardless of particular differences, there is a general tension between *correctness* and *performance*. The correctness of a ML 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?* As with other approximate computing problems, ML can relax its demands on accuracy in order to achieve increases in efficiency. In fact, this relaxation is a requirement in many learning domains. Without it, inference computations can be so inefficient to perform that they become intractable. We describe five such cases below.

3.1 Data Subsampling

Performance directly relates to the size of the task on which we perform learning. Intuitively, if a learning algorithm is slow on small tasks—that is, tasks with small datasets—then that algorithm will be slow, if not computationally intractable, on much larger ones. More concretely, 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. As we show in Figure 2 for the Gradient Descent (GD) algorithm, for larger datasets this summation becomes increasingly costly.

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.,

[4] There have been similar findings in other cities such as Detroit [54].

[5] While this observation speaks to the trade-off between accuracy and efficiency, we do not intend for this to be taken as an endorsement for using risk assessment tools in criminal law domains. We instead apply this example narrowly to explicate accuracy-efficiency considerations, without commenting on the normative implications of what accuracy means in this context or the desirability of the use of such tools.

[6] While "precision" and "accuracy" are different, there is a relationship between them. For our purposes, it useful to think of the degree of precision as a mechanism for controlling how much accuracy is possible to achieve when performing a computation. For example, using fewer bits (i.e, low bit precision) to represent numbers can drastically effect the degree of accuracy of calculations done with those numbers, since this is effectively the same as doing computation on (potentially very highly) rounded numbers (Section 3.2).

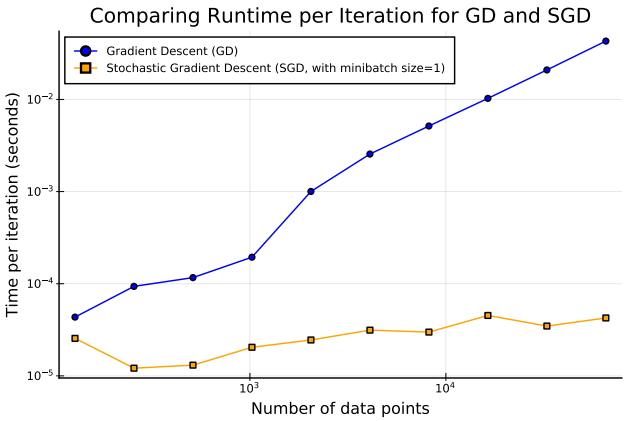

Figure 2. The runtime for GD, a full-batch method (which uses the whole dataset to compute gradients), is coupled with the size of the dataset: as the dataset size increases, so does the runtime per iteration of the algorithm. In contrast, a subsampled, minibatch method like SGD (which here uses only 1 data point to compute the gradient) is decoupled from the dataset size: it maintains a relatively constant runtime per iteration.

the whole dataset) we use a randomly sampled subset of the data points, which entails spending less time on computation. Stochastic Gradient Descent (SGD) is an example of an algorithm that takes this approach. Using a minibatch can often 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 (Figure 2); but, when run for lots of iterations, the final result can still be 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 [13, 49].

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.

3.2 Low-Precision Computation

Another common approach involves using low-precision representations of the numerical values on which the computer performs computations. 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—that we discussed in Section 2.2. Computing with more precise floating-point numbers is more computationally expensive; it tends to take more time (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 [4, 23, 25, 36, 37, 39]. As with the minibatching example in Section 3.1, this sacrifice in accuracy does not necessarily require sacrificing overall correctness if in expectation the algorithm can still theoretically guarantee learning the right distribution. There is also a spectrum at play here; similar to varying the minibatch size to tune the trade-off between accuracy and efficiency, 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 creates [72].

3.3 Resource-Constrained Machine Learning

The prior examples discuss the cost of running computations. Specifically, they discuss how differently-sized batches of data (Section 3.1) and how differing degrees of numerical precision (Section 3.2) directly impact how long it takes a computer to execute a computation. Even though these examples concern a computer's behavior, we have not yet considered how hardware specifications of the computer running the algorithm might 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.

Recent years have seen an increase in the variety of computational devices available and a corresponding increase in the variety of computations we wish to run on them. For example, Internet of Things (IoT) devices and sensors, such as Google Home or Amazon Echo, perform inference. They serve up answers to spoken language questions; however, they also have 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.

Often, these devices can communicate with more sophisticated computers over the Internet, offloading computation or storage to those computers. However, this communication exposes another trade-off between accuracy and efficiency; it takes time to send the data to a remote computer, perform some computation, and then return a response to the device [11]. That computation may be more accurate, 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 very accurate.

Prior work considers how computer vision models can be learned and stored on a mobile device like a smartphone.[7] For such resource-constrained devices, different applications have different needs in terms of how to trade-off between how accurately and how quickly a computation is performed. Some prior work has explored these application-specific needs, providing an interface for flexibly implementing different points along the accuracy-efficiency trade-off spectrum. For example, MobileNets contains manually-tunable parameters that allow the model developer to strike the right balance for particular learning problems [46]. Depending on the application domain, the developer can tune a larger model that uses more resources (i.e., a model that is slower but more accurate) or one that is smaller and uses fewer resources (i.e., a model that is faster but less accurate).

3.4 Markov chain Monte Carlo Sampling

We now delve into a slightly more sophisticated example. We consider a branch of ML that has recently proven particularly useful in the probabilistic modeling of data for Bayesian inference: Markov chain Monte Carlo (MCMC) sampling. To understand MCMC, it's first important to have an intuition regarding how sampling works. We will explain sampling by way of a simple, familiar example: flipping a coin.

When flipping a normal coin, it can either result in "heads" or "tails," with a 50% chance for yielding each. Let us consider that it is possible for a coin to be biased—that the coin is weighted in such a way that, when flipped, it yields heads 60% of the time. In order to figure out how biased the coin is—the probability it yields heads—we flip the coin repeatedly to generate samples of the coin's behavior and record the results. That is, we flip the coin for multiple trials, and after each trial we update the estimated probability that the coin yields a heads result. We can view this updating probability as the information we are learning—we are generating a model of the coin's behavior, which we store as the probability of flipping heads. When we begin flipping the coin, there are not many generated samples. As a result, as shown in Figure 3 our estimation of the probability of heads might change a lot; it can update fairly erratically. Over time, as we generate more samples, the probability estimate becomes more stable. We converge to a probability that does not change very much, giving us a fairly good estimate of the coin's bias.

MCMC can be thought of as a more complicated instantiation of a sampling method like this. Instead of learning the probability of a biased coin, we are trying to learn the parameters of a desired probability distribution. To do this, we construct a Markov chain, from which we iteratively produce samples. Similar to updating the estimated bias of

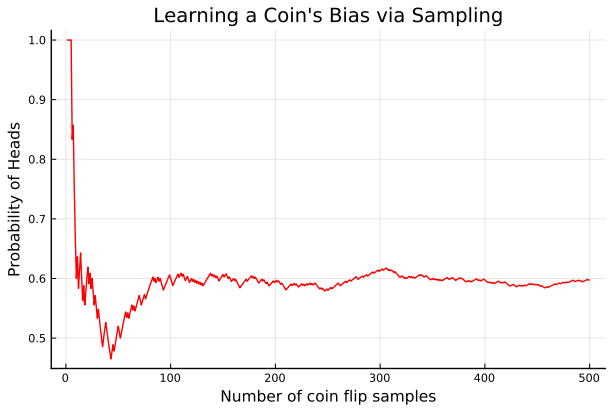

Figure 3. Generating samples of a coin flip to determining a coin's bias (in this case, 60% heads).

the coin after each sampling iteration, in each iteration of MCMC we update our estimation of the distribution parameters. Eventually, the values of the parameters become stable; we converge to an equilibrium in which the samples we continue to draw reflect the desired distribution [15].

While this technique is very powerful for accurately performing Bayesian inference, it comes with significant performance drawbacks. In particular, when the learning problem's dataset is large, the performance of an MCMC algorithm often suffers. Just as in Section 3.1, efficiency and scalability become limited due to computations that require summing over every data point in the dataset; performance is therefore (loosely speaking) inversely proportional to the size of the dataset. Additionally, just as before, we can lessen these limitations by introducing subsampling—by using a randomly selected minibatch of data instead of the whole dataset. However, as we have seen with the accuracy-efficiency trade-off, there is no free lunch; the speed-ups from minibatching can introduce inaccuracy. More specifically, we can lose the guarantee of converging to the correct desired distribution, which can yield potentially disastrous inference results [94]. Instead of yielding *exact* results, the randomness from using minibatches can introduce bias that entails *inexact* results.

Prior work makes the case that inexactness can be worth its performance gains—that it is better to be faster even if there is a risk of losing accuracy, since it can enable scaling up MCMC to big data inference problems. As a result, there is a rich scholarly literature concerning inexact minibatch MCMC methods [18, 50, 77]. However, in practice, data science practitioners often do not use inexact methods; for reliability, they find that it is better to be slow and correct than fast and wrong. Recent work therefore attempts to construct new minibatch MCMC methods that retain exactness—methods that have theoretical guarantees regarding accuracy while also incorporating certain tricks and statistical insights that enable preserving some of the speed-ups minibatching

[7] Aside from being faster, there are several reasons why such local computation and storage might be desirable for a mobile application, as opposed to communicating with and 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.

provides [22, 59, 93–95]. In other words, these exact methods lean toward the accuracy side of the accuracy-efficiency trade-off; they guarantee converging to the correct, desired distribution, but to do so they sacrifice speed in relation to their inexact counterparts, particularly on some types of learning tasks.

3.5 Asynchronous ML

Finally, we examine the trade-off in machine learning in *asynchronous* settings. The examples we have discussed so far are *synchronous*: there is one computer process[8] that does all of the computation, one step at a time. In contrast, it is possible to run computations *asynchronously*, in which different computer processes or threads[9] perform computations side-by-side. This facilitates dividing computationally intensive tasks into parts, such that different portions can happen in parallel and then can be combined to compute the final result.

In other words, the parallelization from asynchrony can lead to speed-ups in ML since multiple parts of the learning problem can be computed at once. However, depending on how the parallel results are combined, it can also lead to decreases in accuracy. That is, if different processes end up working on the overlapping parts of the overarching computation, the process that finishes its computation second can overwrite the value computed by the one that finished first, causing inaccuracies in the results [5, 25, 56, 66]. This can be avoided by forcing the different processes to coordinate their updates, such that they do not overwrite each other. However, such coordination takes time; it enables more accuracy, but decreases efficiency. In some cases, this overwriting is worth the speed-ups it enables; it is still possible to compute good quality learning estimates [24, 73].[10]

So far, our discussion does not take into consideration how the accuracy-efficiency trade-off behaves for ML in real-world, deployed systems—systems that often consist of multiple computers that communicate and work together to solve large, complex problems. Such systems often communicate asynchronously: instead of one computer doing multiple sub-computations at the same time (Section 3.5), there are multiple computers operating in parallel on the same problem. In the next section, we discuss how such real-world distributed ML systems raise unique concerns with regard to accuracy and efficiency.

4 Making Sense of Additional Trade-Offs in Real-World ML Systems

Our overarching aim is to understand the particular tensions between accuracy and efficiency for *distributed machine learning systems*, and how these tensions differ from those we discussed regarding machine learning algorithms in Section 3. To make these distinctions clear, we first clarify some key ideas from distributed computing in Section 4.1.[11] From this basis, we can then layer on more complexity in Section 4.2. We weave in our understanding of the accuracy-efficiency trade-off for ML algorithms from Section 3 and observe how the different tensions interact with each other. Considered together, we demonstrate how machine learning and distributed systems trade-offs present especially challenging problems for real-time, high-impact systems like autonomous vehicles. These real-time domains inform our policy discussion in Section 5.

4.1 Accuracy and Efficiency in Distributed Systems

In contrast with a single, solitary computer, a *distributed system* is a network of computers that communicate with each other. Via this communication, the computers can work together to solve problems. Each computer in the network has its own data and performs its own computations, and it shares data and computation results with other computers in the network when necessary. For example, if a computer needs data from another computer in order to execute a computation, it can request the data from that computer.

Because the computers are in distributed locations and need to communicate, there are important considerations with regard to how efficiently information can be shared between them. That is, when a computer contacts another in the system to request its data, it takes time to complete the request and receive the data—in direct opposition to efficiency. There are also issues of accuracy between computers in the system. Each computer has its own data—its own snapshot of what it knows to be the state of the overarching system. However, that information is not complete; it is just a subset and can possibly contradict the information that other computers in the system have. Simply put, the computers can be inconsistent with each other.

In other words, in distributed systems we can frame the trade-off between accuracy and efficiency as a tension between *consistency* and *latency*—the speed with which the

[8] A computer can run multiple *processes* at once. Each process is an instance of a running program—this is why one can run both an Internet browser and a text editor at the same time. In other words, processes allow for parallel tasks to run on one computer [6].

[9] A thread is a further mechanism for parallelization on a computer, which operates at the level of a process. That is, 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 of computation.

[10] Asynchrony is complementary to other examples in this section. For example, it can be used in combination with minibatching, low-precision, and in MCMC to implement other types of accuracy-efficiency trade-offs.

[11] We touch on this topic only briefly, since our main focus is the behavior of such systems in the context of *machine learning* For more detailed treatment, see Cooper [20] and Cooper and Levy [21].

system updates. 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 [2, 14]. Due to this trade-off, in distributed systems that update their data frequently it is actually quite difficult to quickly build a consistent, holistic understanding of the environment across different computers in the network. This is because consistency is a moving target; each computer processes information locally faster than it can share it with the entire network. Given that it takes time to communicate, it is hard for computers to stay completely up to date with each other.

Nevertheless, for the sake of efficiency, individual computers in the system often need to make decisions in the presence of inconsistency. Otherwise, because of the tension between consistency and latency, waiting for complete consistency across computers before a computer could make local changes would bring the entire system to a standstill. Instead, particular distributed system implementations need to answer the question of how much inconsistency and slowness they can each tolerate, which is often application-dependent.

To understand this spectrum, we will consider a few examples of distributed systems that implement the trade-off differently [26, 41]. First, consider a social media website, which has computers hosting its data distributed all over the world. A user visiting the site from a personal device 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 why on some social media sites it is possible to see out-of-order comments on a feed; the site is making a best effort to resolve its current state, which entails aggregating 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 [26, 58, 86]. The system implements this choice via its communication strategy. Rather than contacting every computer in the system to construct a coordinated, consistent picture (which would take a lot of time) 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 [40, 52].

In contrast to an efficiency-favoring social media site, blockchain technology is a distributed system for storing a transaction ledger that favors consistency at the cost of being slow[63]. In short, it is a distributed system where each computer has its own copy of the entire ledger. When a computer wants to add a transaction to the system, it has to broadcast that information to every computer in the network. All of the computers need to agree on the validity of a transaction before it can be included. As a result, the system proceeds in lockstep, only when there is coordinated agreement.[12]

These different implementation choices reflect different design goals. The cloud was designed for e-commerce applications, in which supplying (even potentially inaccurate) responses quickly to the user is critical for user engagement [9, 16]. For blockchain systems, consistency is paramount; it is crucial that all of the computers agree with each other about the state of the ledger, because it is this agreement that facilitates its reliability as a transaction record.

While these two examples seem to imply that there is an all-or-nothing choice in the trade-off between consistency and latency in distributed systems, this is not the case. Like accuracy and efficiency more generally in approximate computing (Section 2.2), the trade-off between consistency and latency is a spectrum [2, 91]. It is possible to quantify consistency and to measure and monitor its maintenance throughout a distributed system [58, 78]. Developers can reason about the degree of inconsistency their particular system can tolerate safely, and can detect and tune the system's implementation accordingly to also enforce an upper bound on latency [10, 29, 35, 71, 83, 92].

4.2 Bringing it All Together: Trade-Offs in Distributed Machine Learning Systems

Given this background on how accuracy and efficiency are in tension with each other in distributed systems in general (Section 4.1) and our earlier discussion of accuracy-efficiency trade-offs in ML (Section 3), we can now specifically consider real-time (i.e., latency-critical) distributed ML systems. As an example, consider a distributed system of autonomous vehicles. Numerous vehicles are potentially networked together and with other devices, such as smart traffic lights. Moreover, while each vehicle moves throughout the environment with its own local notion of the state of the environment, information that other vehicles possess could also prove useful. For example, if an accident is up ahead, a vehicle closer to the crash can communicate that information to the vehicles behind it, which in turn can apply pressure to their brakes and potentially prevent a pile-up.

In such real-time transportation domains, accuracy and efficiency[13] are both critical. Some ML inference applications may be error tolerant, but in high-stakes domains this may not always be the case; it is unclear how much inaccuracy will be tolerable while still ensuring safety.[14] The way such

[12]This is a tremendous oversimplification for brevity, since the point of introducing this example is to explain trade-offs between accuracy and efficiency. A more detailed treatment appears in Narayanan et al. [64].
[13]For consistency with the main framing of the trade-off between accuracy and efficiency in this paper, we will use this language, instead of consistency and latency, going forward. However, as noted in Section 4.1, consistency and latency can be viewed as cases of accuracy and efficiency, respectively.
[14]For more on the normative values at play in such situations, please refer to Cooper and Levy [21].

systems will need to treat efficiency is similar. They will need to make decisions quickly and, much like the non-computing examples in Section 2.1, 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 [2, 14]. What is different here is the degree of efficiency needed—in some cases, inference decisions will be necessary at subsecond speeds.

In short, it is not entirely clear what the right design goal is for real-time systems like autonomous vehicles and how the trade-off should be implemented for them [27]. Given the dynamic nature of the environment, the particular trade-off implementation may depend on context. Some environments 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. In other cases, having an accurate sense of the environment may be more important than allowing the cars to operate quickly. 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 if the hole is deep enough to warrant veering off the road to avoid it.

Distributed ML systems raise different accuracy-efficiency questions than either distributed systems that do not involve ML, or ML systems that are not distributed. With regard to the former, the kinds of coordination and consistency issues that distributed systems can tolerate while maintaining correctness are different in nature than what newer ML systems can tolerate (particularly around issues like numerical error and staleness) [9, 26, 92]. With regard to the latter, as we saw in Section 3, since ML models (necessarily imperfectly) approximate representations of the world, it is possible for ML models 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 when operating on accurate data. We can extend such data inaccuracies beyond things like subsampling to include data staleness inherent in asynchronous distributed settings. Allowing for such staleness comes with the benefit of increasing efficiency, as the system would not need to wait to synchronize—to completely resolve staleness issues before proceeding with its computation. Similar to the single computer case, their overall output still *can be* correct even when operating on numerically imprecise or stale data in a distributed setting; however, existing work in this field does not necessarily guarantee such output *must be* correct [5, 30, 36, 56, 66, 74, 96].

Instead, prior work has examined this phenomenon at a high level by looking at the correctness and the performance of end-to-end ML systems, rather than directly evaluating the underlying accuracy-efficiency trade-off. This work focuses on empirical results for tuning the staleness of the underlying data storage layer. Tuning has generally either been manual—curated to the particular problem domain—or absent, leaving the user to pick from a few predefined settings that enforce high accuracy, ignore accuracy altogether for efficiency, or attempt some middle-ground, "in-between" approach [3, 44, 51, 55, 69]. Attempts at more flexible trade-offs have entailed very domain- or algorithm- specific solutions [57, 68, 88].

While it is possible to implement any of these different points in the trade-off, current large-scale systems for distributed learning and inference tend to opt for efficiency. They focus on minimizing communication between computers in the system in order to be efficient enough to scale to larger problems. Some of these systems can achieve orders of magnitude in performance improvements by dropping updates without simultaneously destroying correctness [66, 84]; however, it is not clear these approaches will work for real-time distributed ML systems that are safety-critical, such as autonomous vehicles. It will not always be feasible 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 then impacts the other side of the accuracy-efficiency trade-off, leading to inefficiencies from bottlenecks in coordination between computers.[15]

5 A Call to Action: Enabling the Regulation of the Accuracy-Efficiency Trade-Off

We have taken considerable space to clarify a variety of accuracy-efficiency trade-offs—from how they generally impact the field of computing to how they describe the range of possible behaviors for distributed machine learning systems.

More specifically, it is necessary and urgent to expose the accuracy-efficiency trade-off because it is a potential lever for regulation. Though various manifestations of the trade-off are well-acknowledged in technical communities, they have not, to date, been legible to policymakers. We argue that policymakers must understand the implications of the accuracy-efficiency trade-off in order to responsibly regulate emerging technologies—that it is necessary and urgent to expose the trade-off as a potential lever for regulation.

As we have documented in Sections 2.2-4.2, this trade-off is not binary; it is a spectrum and can be treated like a tunable dial set appropriately to the context. Our hope is that exposing this dial will provide a certain degree of technical transparency to lawmakers, such that high-stakes systems do not get deployed without sufficient public oversight [20, 21]. Contemporary policy debates about high-stakes, time-sensitive machine learning applications—in domains

[15] 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" [8, 28, 34, 42, 89].

like policing, warfare, and public health—often involve concerns about what degree of accuracy we ought to demand from such systems. These concerns often arise in the course of attempting to minimize disparate outcomes across groups (e.g., differential accuracy rates for face recognition along dimensions of race and gender [17]). But debates about the harms of inaccuracy are incomplete if they fail to acknowledge and reckon with the technical trade-off between accuracy and efficiency. Accuracy may necessarily be limited when speed is essential, and as we have seen, the speed of decision-making can implicate important public values as well [21]. Informed policy debate about machine learning must pay attention to the limits imposed by this trade-off.

Beyond exposing this trade-off, we also propose a twofold call to action. The first portion of this call is for computer scientists. While our work here exposes the trade-off between accuracy and efficiency and how to engage with it—to build systems that can prioritize application-dependent balances between the two—it also indicates gaps in existing approaches in real-time ML systems. These gaps imply that existing systems will likely not suffice for high-stakes, emerging applications such as autonomous vehicles. In particular, in Section 4.2 we explain the importance of needing to make the accuracy-efficiency trade-off transparent in a system's implementation; a system's ability to be assessed with regard to this trade-off should be considered as important as every other technical feature. A potential future research direction could mathematically formalize the semantics of the trade-off in ML systems. This could enable building tools to optimally tune the trade-off between consistency and latency for different classes of distributed ML algorithms, balancing their individual accuracy and efficiency needs.

Such tools would also provide policymakers with insight into how certain implementation decisions impact overall system behavior. This is crucial because, as we have shown throughout this paper, low-level technical decisions are not trivial; they should not be dismissed as "just implementation details" left up to the whims of engineers without public oversight. To be clear, we are not claiming that policymakers need to understand the full extent of low-level technical details to provide this oversight. Rather, we are suggesting that surfacing the higher-level trade-offs that lower-level decisions entail clarifies valid sites for potential policy intervention [21, 47, 62]. One can then think of such trade-offs as the right layer of abstraction with which policymakers can engage. At this level, policymakers can reason about the normative values and policy goals implicated by the trade-off in different domains [21, 31, 32]. The case of the accuracy-efficiency trade-off, for example, can be used to clarify how lower-level engineering decisions relate to notions of safety and quality [76].

It is this reasoning that informs the second part of our call to action: policymakers should view the accuracy-efficiency trade-off as a regulable decision point at which they can meaningfully intervene. They can use these trade-offs to assess the expected behavior of real-time ML systems. As a result, we can fairly pose to policymakers questions like the following: At what point should we deem information of sufficiently high quality to justify the execution of potentially high-impact decisions by technical systems? When is it safe for a system to spend more time computing inference outcomes, particularly when more efficient heuristics do not sufficiently remove uncertainty from automated decision-making?

In other words, by giving policymakers the tools to reason about these higher-level trade-offs, we are able to take a step toward closing what Jasanoff terms the "responsibility gap." That is, policymakers will have a more sufficient understanding of technology and will be better equipped to gauge the range of possibilities for its governance. This way, when technological failures occur, rather than viewing them simply as "unintended consequences" or "normal accidents" [70], policymakers can more actively participate in the evaluation of how uncertainty in probabilistic, automated decision systems contributes to the construction of risk [43, 47].

6 Conclusion

This two-pronged call to action highlights the relationship between uncertainty and risk in distributed machine learning systems. By providing a mechanism to reason about the accuracy-efficiency trade-off, computer scientists expose a particular kind of decisional uncertainty that depends on time [12, 45]. Clarifying this uncertainty does not, however, identify specific risks that these automated decisions bring about. Given the uncertainties involved, it is up to regulators to frame potential risks and to identify the normative, domain-specific values at play [47]. That is, while computer scientists can reason about how much error is tolerable due to the accuracy-efficiency trade-off (Section 2.2), we contend that policymakers and regulators need to determine how much of the resulting risk is tolerable.

In select cases, in which it is possible to deem the amount of predetermined risk to be intolerable, policymakers could disallow particular technical systems from widespread deployment [27, 70, 75]. However, in most cases, it may not be possible to preemptively fully analyze the risk landscape [79, 82]. Instead, this is where exposing the trade-off between accuracy and efficiency can lead to accountability after-the-fact [7, 38, 53]. In other words, when deployed in the wild for long enough, due to their complexity real-time, high-stakes ML systems are likely to incur harm [65, 70, 85]. Given that this is unavoidable, it is important to build tools like those we call for in Section 5. This way, it will be possible to determine if a system has deviated further than expected from its normal (what we consider to be acceptable) behavior [76]—cases in which policymakers and regulators can hold the appropriate stakeholders to account.

Acknowledgments

This work was made possible by generous funding from Adrian Sampson and the John D. and Catherine T. MacArthur Foundation. We would like to thank Jaime Ashander, Ken Birman, Em Feder Cooper, Thomas G. Dietterich, James Grimmelmann, Ido Kilovaty, Kristian Lum, Alan Mackworth, Helen Nissenbaum, Alec Pollak, and Matthew Sun for their comments and suggestions on various versions of this work—with particular appreciation given to Harry Auster for his incisive feedback.

Code

The code used to generate Figures 2 and 3 can be found at https://github.com/pasta41/lml-2020.

References

[1] 2020. Amazon S3 Storage Classes. https://aws.amazon.com/s3/storage-classes/

[2] Daniel Abadi. 2012. Consistency Tradeoffs in Modern Distributed Database System Design: CAP is Only Part of the Story. *Computer* 45, 2 (Feb. 2012), 37–42.

[3] 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. 2016. TensorFlow: A System for Large-scale Machine Learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (Savannah, GA, USA) *(OSDI'16)*. USENIX Association, Berkeley, CA, USA, 265–283.

[4] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. 2017. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 1709–1720.

[5] Dan Alistarh, Christopher De Sa, and Nikola Konstantinov. 2018. The Convergence of Stochastic Gradient Descent in Asynchronous Shared Memory. *CoRR* abs/1803.08841 (2018). arXiv:1803.08841

[6] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. 2018. *Operating Systems: Three Easy Pieces* (1.00 ed.). Arpaci-Dusseau Books.

[7] Michael Backes, Peter Druschel, Andreas Haeberlen, and Dominique Unruh. 2009. CSAR: A Practical and Provable Technique to Make Randomized Systems Accountable. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2009, San Diego, California, USA, 8th February - 11th February 2009*.

[8] B. R. Badrinath and Krithi Ramamritham. 1992. Semantics-based Concurrency Control: Beyond Commutativity. *ACM Trans. Database Syst.* 17, 1 (March 1992), 163–199.

[9] Peter Bailis, Shivaram Venkataraman, Michael J. Franklin, Joseph M. Hellerstein, and Ion Stoica. 2012. Probabilistically Bounded Staleness for Practical Partial Quorums. *Proc. VLDB Endow.* 5, 8 (April 2012), 776–787.

[10] D. Barbara and H. Garcia-Molina. 1990. The case for controlled inconsistency in replicated data. In *[1990] Proceedings. Workshop on the Management of Replicated Data*. 35–38.

[11] Ken Birman, Bharath Hariharan, and Christopher De Sa. 2019. Cloud-Hosted Intelligence for Real-Time IoT Applications. *SIGOPS Oper. Syst. Rev.* 53, 1 (July 2019), 7–13.

[12] Mark Boddy and Thomas L. Dean. 1994. Deliberation Scheduling for Problem Solving in Time-Constrained Environments. *Artif. Intell.* 67, 2 (June 1994), 245–285.

[13] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. 2018. Optimization Methods for Large-Scale Machine Learning. *SIAM Rev.* 60, 2 (Jan 2018), 223–311.

[14] Eric Brewer. 2012. CAP Twelve Years Later: How the "Rules" Have Changed. *Computer* 45, 2 (2012), 23–29.

[15] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. 2011. *Handbook of Markov Chain Monte Carlo*. CRC Press.

[16] Jake Brutlag. 2009. Speed matters for google web search.

[17] Joy Buolamwini and Timnit Gebru. 2018. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency (Proceedings of Machine Learning Research, Vol. 81)*, Sorelle A. Friedler and Christo Wilson (Eds.). PMLR, New York, NY, USA, 77–91.

[18] Tianqi Chen, Emily Fox, and Carlos Guestrin. 2014. Stochastic gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*. 1683–1691.

[19] Alexandra Chouldechova. 2017. Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments. *Big Data* 5, 2 (Jun 2017), 153–163.

[20] A. Feder Cooper. 2018. Imperfection is the Norm: A Computer Systems Perspective on IoT and Enforcement. (2018). https://law.yale.edu/isp/events/imperfect-enforcement (Im)Perfect Enforcement Conference.

[21] A. Feder Cooper and Karen Levy. 2020. Distributing Accountability and Distributed Computing: Policy Implications in Real-Time Computer Systems. (2020). Under submission.

[22] Robert Cornish, Paul Vanetti, Alexandre Bouchard-Côté, George Deligiannidis, and Arnaud Doucet. 2019. Scalable Metropolis-Hastings for exact Bayesian inference with large datasets. *International Conference on Machine Learning* (2019).

[23] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. arXiv:1511.00363 [cs.LG]

[24] Constantinos Daskalakis, Nishanth Dikkala, and Siddhartha Jayanti. 2018. HOGWILD!-Gibbs Can Be PanAccurate. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS '18)*. Curran Associates Inc., Red Hook, NY, USA, 32–41.

[25] Christopher De Sa, Matthew Feldman, Christopher Ré, and Kunle Olukotun. 2017. Understanding and Optimizing Asynchronous Low-Precision Stochastic Gradient Descent. In *Proceedings of the 44th Annual International Symposium on Computer Architecture* (Toronto, ON, Canada) *(ISCA '17)*. Association for Computing Machinery, New York, NY, USA, 561–574.

[26] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. 2007. Dynamo: Amazon's Highly Available Key-value Store. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles* (Stevenson, Washington, USA) *(SOSP '07)*. ACM, New York, NY, USA, 205–220.

[27] Thomas G. Dietterich. 2018. Robust artificial intelligence and robust human organizations. *Frontiers of Computer Science* 13, 1 (Dec 2018), 1–3.

[28] Lisa Cingiser DiPippo and Victor Fay Wolfe. 1997. Object-Based Semantic Real-Time Concurrency Control with Bounded Imprecision. *IEEE Trans. on Knowl. and Data Eng.* 9, 1 (Jan. 1997), 135–147.

[29] W. Du and A. Elmagarmid. 1989. Quasi Serializability: A Correctness Criterion for Global Concurrency Control in InterBase. In *Proceedings of the 15th International Conference on Very Large Data Bases* (Amsterdam, The Netherlands) *(VLDB '89)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 347–355.

[30] Sanghamitra Dutta, Gauri Joshi, Soumyadip Ghosh, Parijat Dube, and Priya Nagpurkar. 2018. Slow and Stale Gradients Can Win the Race: Error-Runtime Trade-offs in Distributed SGD. arXiv:1803.01113 [stat.ML]

[31] M. Flanagan, Daniel Howe, and H. Nissenbaum. 2008. Embodying values in technology: Theory and practice. *Information Technology and Moral Philosophy* (01 2008), 322–353.

[32] Batya Friedman and David G. Hendry. 2019. *Value Sensitive Design: Shaping Technology with Moral Imagination.* The MIT Press.

[33] Batya Friedman and Helen Nissenbaum. 1996. Bias in Computer Systems. *ACM Trans. Inf. Syst.* 14, 3 (July 1996), 330–347.

[34] Hector Garcia-Molina. 1983. Using Semantic Knowledge for Transaction Processing in a Distributed Database. *ACM Trans. Database Syst.* 8, 2 (June 1983), 186–213.

[35] Wojciech Golab, Xiaozhou Li, and Mehul A. Shah. 2011. Analyzing Consistency Properties for Fun and Profit. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing* (San Jose, California, USA) *(PODC '11)*. ACM, New York, NY, USA, 197–206.

[36] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. 2014. Compressing Deep Convolutional Networks using Vector Quantization. *CoRR* abs/1412.6115 (2014). arXiv:1412.6115

[37] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep Learning with Limited Numerical Precision. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37* (Lille, France) *(ICML'15)*. JMLR.org, 1737–1746.

[38] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. 2007. PeerReview: Practical Accountability for Distributed Systems. *SIGOPS Oper. Syst. Rev.* 41, 6 (Oct. 2007), 175–188.

[39] Song Han, Huizi Mao, and William J. Dally. 2015. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. arXiv:1510.00149 [cs.CV]

[40] Joseph M. Hellerstein and Peter Alvaro. 2019. Keeping CALM: When Distributed Consistency is Easy. *CoRR* abs/1901.01930 (2019). arXiv:1901.01930

[41] M. Herlihy. 1990. Apologizing Versus Asking Permission: Optimistic Concurrency Control for Abstract Data Types. *ACM Trans. Database Syst.* 15, 1 (March 1990), 96–124.

[42] Nathaniel Herman, Jeevana Priya Inala, Yihe Huang, Lillian Tsai, Eddie Kohler, Barbara Liskov, and Liuba Shrira. 2016. Type-aware Transactions for Faster Concurrent Code. In *Proceedings of the Eleventh European Conference on Computer Systems* (London, United Kingdom) *(EuroSys '16)*. ACM, New York, NY, USA, Article 31, 16 pages.

[43] Stephen Hilgartner. 1992. The Social Construction of Risk Objects: Or, How to Pry Open Networks of Risk. In *Organizations, Uncertainties, and Risk*, James F. Short and Lee Clark (Eds.). 39–53.

[44] Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B. Gibbons, Garth A Gibson, Greg Ganger, and Eric P Xing. 2013. More Effective Distributed ML via a Stale Synchronous Parallel Parameter Server. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1223–1231.

[45] Eric J. Horvitz. 1987. Reasoning about Beliefs and Actions under Computational Resource Constraints. In *Proceedings of the Third Conference on Uncertainty in Artificial Intelligence* (Seattle, WA) *(UAI '87)*. AUAI Press, Arlington, Virginia, USA, 429–447.

[46] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861 [cs.CV]

[47] Sheila Jasanoff. 2016. *The Ethics of Invention: Technology and the Human Future.* New York: W.W. Norton & Company.

[48] Daniel Kahneman, Paul Slovic, and Amos Tversky. 1982. *Judgment under Uncertainty: Heuristics and Biases.* New York: Cambridge University Press.

[49] Jürgen Kiefer and Jacob Wolfowitz. 1952. Stochastic Estimation of the Maximum of a Regression Function.

[50] Anoop Korattikara, Yutian Chen, and Max Welling. 2014. Austerity in MCMC land: Cutting the Metropolis-Hastings budget. In *International Conference on Machine Learning*. 181–189.

[51] Jack Kosaian, K. V. Rashmi, and Shivaram Venkataraman. 2019. Parity Models: Erasure-Coded Resilience for Prediction Serving Systems. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles* (Huntsville, Ontario, Canada) *(SOSP '19)*. Association for Computing Machinery, New York, NY, USA, 30–46.

[52] Sudha Krishnamurthy, William Sanders, and Michel Cukier. 2002. An Adaptive Framework for Tunable Consistency and Timeliness Using Replication. (05 2002).

[53] B. W. Lampson. 2004. Computer Security in the Real World. *Computer* 37, 6 (June 2004), 37–46.

[54] Timothy B. Lee. 2020. Detroit police chief cops to 96-percent facial recognition error rate. *Ars Technica* (June 2020). https://arstechnica.com/tech-policy/2020/06/detroit-police-chief-admits-facial-recognition-is-wrong-96-of-the-time

[55] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. 2014. Scaling Distributed Machine Learning with the Parameter Server. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation* (Broomfield, CO) *(OSDI'14)*. USENIX Association, Berkeley, CA, USA, 583–598.

[56] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. 2017. Asynchronous Decentralized Parallel Stochastic Gradient Descent. arXiv:1710.06952 [math.OC]

[57] Wyatt Lloyd, Michael J. Freedman, Michael Kaminsky, and David G. Andersen. 2011. Don'T Settle for Eventual: Scalable Causal Consistency for Wide-area Storage with COPS. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles* (Cascais, Portugal) *(SOSP '11)*. ACM, New York, NY, USA, 401–416.

[58] Haonan Lu, Kaushik Veeraraghavan, Philippe Ajoux, Jim Hunt, Yee Jiun Song, Wendy Tobagus, Sanjeev Kumar, and Wyatt Lloyd. 2015. Existential Consistency: Measuring and Understanding Consistency at Facebook. In *Proceedings of the 25th Symposium on Operating Systems Principles* (Monterey, California) *(SOSP '15)*. ACM, New York, NY, USA, 295–310.

[59] Dougal Maclaurin and Ryan Prescott Adams. 2015. Firefly Monte Carlo: Exact MCMC with subsets of data. In *Twenty-Fourth International Joint Conference on Artificial Intelligence.*

[60] Sparsh Mittal. 2016. A Survey of Techniques for Approximate Computing. *ACM Comput. Surv.* 48, 4, Article 62 (March 2016), 33 pages.

[61] Thierry Moreau, Joshua San Miguel, Mark Wyse, James Bornholt, Armin Alaghi, Luis Ceze, Natalie Enright Jerger, and Adrian Sampson. 2018. A Taxonomy of General Purpose Approximate Computing Techniques. *IEEE Embed. Syst. Lett.* 10, 1 (March 2018), 2–5.

[62] D.K. Mulligan and K.A. Bamberger. 2018. Saving governance-by-design. *California Law Review* 106 (06 2018), 697–784.

[63] Satoshi Nakamoto. 2009. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf

[64] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. 2016. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction.* Princeton University Press, USA.

[65] Helen Nissenbaum. 1996. Accountability in a Computerized Society. *Science and Engineering Ethics* 2 (1996), 25–42.

[66] Feng Niu, Benjamin Recht, Christopher Re, and Stephen J. Wright. 2011. HOGWILD!: A Lock-free Approach to Parallelizing Stochastic Gradient Descent. In *Proceedings of the 24th International Conference on Neural Information Processing Systems* (Granada, Spain) *(NIPS'11)*.

Curran Associates Inc., USA, 693–701.

[67] Paul Ohm and Jonathan Frankle. 2019. Desirable Inefficiency. In *Florida Law Review*, Vol. 70. 777–836. Issue 4.

[68] Xinghao Pan, Joseph Gonzalez, Stefanie Jegelka, Tamara Broderick, and Michael I. Jordan. 2013. Optimistic Concurrency Control for Distributed Unsupervised Learning. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1* (Lake Tahoe, Nevada) *(NIPS'13)*. Curran Associates Inc., USA, 1403–1411.

[69] Xinghao Pan, Maximilian Lam, Stephen Tu, Dimitris Papailiopoulos, Ce Zhang, Michael I Jordan, Kannan Ramchandran, and Christopher Ré. 2016. Cyclades: Conflict-free Asynchronous Machine Learning. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 2568–2576.

[70] Charles Perrow. 1999. *Normal Accidents: Living with High Risk Technologies - Updated Edition.* Princeton University Press, Princeton, New Jersey.

[71] K. Ramamritham and C. Pu. 1995. A formal characterization of epsilon serializability. *IEEE Transactions on Knowledge and Data Engineering* 7, 6 (1995), 997–1007.

[72] Christopher De Sa, Megan Leszczynski, Jian Zhang, Alana Marzoev, Christopher R. Aberger, Kunle Olukotun, and Christopher Ré. 2018. High-Accuracy Low-Precision Training. (2018). arXiv:1803.03383

[73] Christopher De Sa, Chris Re, and Kunle Olukotun. 2016. Ensuring Rapid Mixing and Low Bias for Asynchronous Gibbs Sampling. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 1567–1576.

[74] Christopher De Sa, Ce Zhang, Kunle Olukotun, and Christopher Ré. 2015. Taming the Wild: A Unified Analysis of Hogwild!-Style Algorithms. *CoRR* abs/1506.06438 (2015). arXiv:1506.06438

[75] Noah Sachs. 2011. Rescuing the Strong Precautionary Principle from its Critics. *University of Illinois Law Review* 2011 (2011), 1285–1338.

[76] Adrian Sampson. 2015. *Hardware and Software for Approximate Computing*. Ph.D. Dissertation. University of Washington. https://www.cs.cornell.edu/~asampson/media/dissertation.pdf

[77] Daniel Seita, Xinlei Pan, Haoyu Chen, and John Canny. 2016. An efficient minibatch acceptance test for Metropolis-Hastings. (2016). arXiv:1610.06848

[78] Zechao Shang, Jeffrey Xu Yu, and Aaron J. Elmore. 2018. RushMon: Real-time Isolation Anomalies Monitoring. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18)*. ACM, New York, NY, USA, 647–662.

[79] Henry E. Smith. 2015. Equity as Second-Order Law: The Problem of Opportunism. In *Harvard Public Law Working Paper No. 15-13.*

[80] Sonja B. Starr. 2014. Evidence-Based Sentencing and the Scientific Rationalization of Discrimination. *Stanford Law Review* 66 (2014), 803–872.

[81] Cass R. Sunstein. 2002. Hazardous Heuristics. *U Chicago Law & Economics* (2002).

[82] Cass R. Sunstein. 2003. Beyond the Precautionary Principle. *U Chicago Law & Economics* (2003).

[83] Francisco J. Torres-Rojas, Mustaque Ahamad, and Michel Raynal. 1999. Timed Consistency for Shared Distributed Objects. In *PODC*.

[84] J. Tsitsiklis, D. Bertsekas, and M. Athans. 1986. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Trans. Automat. Control* 31, 9 (1986), 803–812.

[85] Diane Vaughan. 1996. *The Challenger launch Decision: Risky Technology, Culture, and Deviance at NASA.* University of Chicago Press.

[86] Werner Vogels. 2009. Eventually Consistent. *Commun. ACM* 52, 1 (Jan. 2009), 40–44.

[87] Carl von Clausewitz. 1832. *Vom Kriege.* Ferdinand Dümmler.

[88] Jinliang Wei, Wei Dai, Aurick Qiao, Qirong Ho, Henggang Cui, Gregory R. Ganger, Phillip B. Gibbons, Garth A. Gibson, and Eric P. Xing. 2015. Managed Communication and Consistency for Fast Data-parallel Iterative Analytics. In *Proceedings of the Sixth ACM Symposium on Cloud Computing* (Kohala Coast, Hawaii) *(SoCC '15)*. ACM, New York, NY, USA, 381–394.

[89] W. E. Weihl. 1988. Commutativity-based concurrency control for abstract data types. *IEEE Trans. Comput.* 37, 12 (1988), 1488–1505.

[90] MC Weinstein, KA Freedberg, EP Hyle, and AD Paltiel. 2020. Waiting for Certainty on Covid-19 Antibody Tests - At What Cost? *New England Journal of Medicine* (2020).

[91] Haifeng Yu and Amin Vahdat. 2000. 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* (San Diego, California) *(OSDI'00)*. USENIX Association, Berkeley, CA, USA, Article 21.

[92] Haifeng Yu and Amin Vahdat. 2000. Efficient Numerical Error Bounding for Replicated Network Services. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB '00)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 123–133.

[93] Ruqi Zhang, A Feder Cooper, and Christopher De Sa. 2020. AMAGOLD: Amortized Metropolis Adjustment for Efficient Stochastic Gradient MCMC. *International Conference on Artificial Intelligence and Statistics* (2020).

[94] Ruqi Zhang, A Feder Cooper, and Christopher De Sa. 2020. Asymptotically Optimal Exact Minibatch Metropolis-Hastings. *ArXiv preprint* (2020).

[95] Ruqi Zhang and Christopher M De Sa. 2019. Poisson-Minibatching for Gibbs Sampling with Convergence Rate Guarantees. In *Advances in Neural Information Processing Systems*. 4923–4932.

[96] Wei Zhang, Suyog Gupta, Xiangru Lian, and Ji Liu. 2015. Staleness-aware Async-SGD for Distributed Deep Learning. *CoRR* abs/1511.05950 (2015). arXiv:1511.05950