A Berkeley View of Systems Challenges for AI

Ion Stoica
Dawn Song
Raluca Ada Popa
David A. Patterson
Michael W. Mahoney
Randy H. Katz
Anthony D. Joseph
Michael Jordan
Joseph M. Hellerstein
Joseph Gonzalez
Ken Goldberg
Ali Ghodsi
David E. Culler
Pieter Abbeel

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2017-159
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-159.html

October 16, 2017

A Berkeley View of Systems Challenges for AI

Ion Stoica, Dawn Song, Raluca Ada Popa, David Patterson, Michael W. Mahoney, Randy Katz, Anthony D. Joseph, Michael Jordan, Joseph M. Hellerstein, Joseph Gonzalez, Ken Goldberg, Ali Ghodsi, David Culler, Pieter Abbeel*

ABSTRACT

With the increasing commoditization of computer vision, speech recognition and machine translation systems and the widespread deployment of learning-based back-end technologies such as digital advertising and intelligent infrastructures, AI (Artificial Intelligence) has moved from research labs to production. These changes have been made possible by unprecedented levels of data and computation, by methodological advances in machine learning, by innovations in systems software and architectures, and by the broad accessibility of these technologies.

The next generation of AI systems promises to accelerate these developments and increasingly impact our lives via frequent interactions and making (often mission-critical) decisions on our behalf, often in highly personalized contexts. Realizing this promise, however, raises daunting challenges. In particular, we need AI systems that make timely and safe decisions in unpredictable environments, that are robust against sophisticated adversaries, and that can process ever increasing amounts of data across organizations and individuals without compromising confidentiality. These challenges will be exacerbated by the end of the Moore's Law, which will constrain the amount of data these technologies can store and process. In this paper, we propose several open research directions in systems, architectures, and security that can address these challenges and help unlock AI's potential to improve lives and society.

KEYWORDS

AI, Machine Learning, Systems, Security

1 INTRODUCTION

Conceived in the early 1960's with the vision of emulating human intelligence, AI has evolved towards a broadly applicable engineering discipline in which algorithms and data are brought together to solve a variety of pattern recognition, learning, and decision-making problems. Increasingly, AI intersects with other engineering and scientific fields and cuts across many disciplines in computing.

In particular, computer *systems* have already proved essential in catalyzing recent progress in AI. Advances in parallel hardware [31, 58, 90] and scalable software systems [32, 46, 114] have sparked the development of new machine learning frameworks [14, 31, 98] and algorithms [18, 56, 62, 91] to allow AI to address large-scale, real-world problems. Rapidly decreasing storage costs [1, 80], crowd-sourcing, mobile applications, internet of things (IoT), and the competitive advantage of data [40] have driven further investment in data-processing systems and AI technologies [87]. The overall effect is that AI-based solutions are beginning to approach or even surpass human-level capabilities in a range of real-world tasks. Maturing AI technologies are not only powering existing industries—including web search, high-speed trading and commerce—but are helping to foster new industries around IoT, augmented reality, biotechnology and autonomous vehicles.

These applications will require AI systems to interact with the real world by making automatic decisions. Examples include autonomous drones, robotic surgery, medical diagnosis and treatment, virtual assistants, and many more. As the real world is continually changing, sometimes unexpectedly, these applications need to support *continual or life-long* learning [96, 109] and *never-ending* learning [76]. Life-long learning systems aim at solving multiple tasks sequentially by efficiently transferring and utilizing knowledge from already learned tasks to new tasks while minimizing the effect of catastrophic forgetting [71]. Never-ending learning is concerned with mastering a set of tasks in each iteration, where the set keeps growing and the performance on all the tasks in the set keeps improving from iteration to iteration.

Meeting these requirements raises daunting challenges, such as active exploration in dynamic environments, secure and robust decision-making in the presence of adversaries or noisy and unforeseen inputs, the ability to explain decisions, and new modular architectures that simplify building such applications. Furthermore, as Moore's Law is ending, one can no longer count on the rapid increase of computation and storage to solve the problems of next-generation AI systems.

Solving these challenges will require synergistic innovations in architecture, software, and algorithms. Rather than addressing specific AI algorithms and techniques, this paper examines the essential role that systems will play in addressing challenges in AI and proposes several promising research directions on that frontier.

2 WHAT IS BEHIND AI'S RECENT SUCCESS

The remarkable progress in AI has been made possible by a "perfect storm" emerging over the past two decades, bringing together: (1) massive amounts of data, (2) scalable computer and software systems, and (3) the broad accessibility of these technologies. These trends have allowed core AI algorithms and architectures, such as deep learning, reinforcement learning, and Bayesian inference to be explored in problem domains of unprecedented scale and scope.

2.1 Big data

With the widespread adoption of online global services, mobile smartphones, and GPS by the end of 1990s, internet companies such as Google, Amazon, Microsoft, and Yahoo! began to amass huge amounts of data in the form of audio, video, text, and user logs. When combined with machine learning algorithms, these massive data sets led to qualitatively better results in a wide range of

* Authors listed in reverse alphabetical order.

core services, including classical problems in information retrieval, information extraction, and advertising [49].

2.2 Big systems

Processing this deluge of data spurred rapid innovations in computer and software systems. To store massive amounts of data, internet service companies began to build massive-scale datacenters, some of which host nearly 100,000 servers, and provide EB [65] of storage. To process this data, companies built new large-scale software systems able to run on clusters of cheap commodity servers. Google developed MapReduce [32] and Google File System [43], followed shortly by the open-source counterpart, Apache Hadoop [7]. Then came a plethora of systems [46, 55, 60, 67, 114], that aimed to improve speed, scale, and ease of use. These hardware and software innovations led to the datacenter becoming the new computer [11].

With the growing demand for machine learning (ML), researchers and practitioners built libraries on top of these systems to satisfy this demand [8, 52, 75].

The recent successes of deep learning (DL) have spurred a new wave of specialized software systems have emerged to scale out these workloads on CPU clusters and take advantage of specialized hardware, such as GPUs and TPUs. Examples include TensorFlow [2], Caffe [57], Chainer [20], PyTorch [89], and MXNet [22].

2.3 Accessibility to state-of-the-art technology

The vast majority of systems that process data and support AI workloads are built as open-source software, including Spark [114], TensorFlow [2], MXNet [22], Caffe [57], PyTorch [89], and BigDL [15]. Open source allows organizations and individuals alike to leverage state-of-the-art software technology without incurring the prohibitive costs of development from scratch or licensing fees.

The wide availability of public cloud services (e.g., AWS, Google Cloud, and MS Azure) allows everyone to access virtually unlimited amounts of processing and storage without needing to build large datacenters. Now, researchers can test their algorithms at a moment's notice on numerous GPUs or FPGAs by spending just a few thousands of dollars, which was unthinkable a decade ago.

3 TRENDS AND CHALLENGES

While AI has already begun to transform many application domains, looking forward, we expect that AI will power a much wider range of services, from health care to transportation, manufacturing to defense, entertainment to energy, and agriculture to retail. Moreover, while large-scale systems and ML frameworks have already played a pivotal role in the recent success of AI, looking forward, we expect that, together with security and hardware architectures, systems will play an even more important role in enabling the broad adoption of AI. To realize this promise, however, we need to address significant challenges that are driven by the following trends.

3.1 Mission-critical AI

With ongoing advances in AI in applications, from banking to autonomous driving to robot-assisted surgery and to home automation, AI is poised to drive more and more mission-critical applications where human well-being and lives are at stake.

As AI will increasingly be deployed in dynamic environments, AI systems will need to continually *adapt* and *learn* new "skills" as the environment changes. For example, a self-driving car could quickly adapt to unexpected and dangerous road conditions (e.g., an accident or oil on the road), by learning in real time from other cars that have successfully dealt with these conditions. Similarly, an AI-powered intrusion-detection system must quickly identify and learn new attack patterns as they happen. In addition, such mission-critical applications must handle noisy inputs and defend against malicious actors.

Challenges: *Design AI systems that learn continually by interacting with a dynamic environment, while making decisions that are timely, robust, and secure.*

3.2 Personalized AI

From virtual assistants to self-driving cars and political campaigns, user-specific decisions that take into account user behavior (e.g., a virtual assistant learning a user's accent) and preferences (e.g., a self-driving system learning the level of "aggressiveness" a user is comfortable with) are increasingly the focus. While such personalized systems and services provide new functionality and significant economic benefits, they require collecting vast quantities of sensitive personal information and their misuse could affect users' economic and psychological wellbeing.

Challenges: *Design AI systems that enable personalized applications and services yet do not compromise users' privacy and security.*

3.3 AI across organizations

Companies are increasingly leveraging third-party data to augment their AI-powered services [27]. Examples include hospitals sharing data to prevent epidemic outbreaks and financial institutions sharing data to improve their fraud-detection capabilities. The proliferation of such applications will lead to a transition from data silos—where one company collects data, processes it, and provides the service—to data ecosystems, where applications learns and make decisions using data owned by different organizations.

Challenges: *Design AI systems that can train on datasets owned by different organizations without compromising their confidentiality, and in the process provide AI capabilities that span the boundaries of potentially competing organization.*

3.4 AI demands outpacing the Moore's Law

The ability to process and store huge amounts of data has been one of the key enablers of the AI's recent successes (see Section 2.1). However, keeping up with the data being generated will become increasingly difficult due to the following two trends.

First, data continues to grow exponentially. A 2015 Cisco white paper [25] claims that the amount of data generated by Internet of Everything (IoE) devices by 2018 to be 400ZB, which is almost 50x the estimated traffic in 2015. According to a recent study [100], by 2025, we will need a three-to-four orders of magnitude improvement in compute throughput to process the aggregate output of all genome sequencers in the world. This would require computation resources to at least double every year.

Second, this explosion of data is coming at a time when our historically rapidly improving hardware technology is coming to a grinding halt [53]. The capacity of DRAMs and disks are expected to double just once in the next decade, and it will take two decades before the performance of CPUs doubles. This slowdown means that storing and processing all generated data will become impracticable.

Challenges: *Develop domain-specific architectures and software systems to address the performance needs of future AI applications in the post-Moore's Law era, including custom chips for AI workloads, edge-cloud systems to efficiently process data at the edge, and techniques for abstracting and sampling data.*

4 RESEARCH OPPORTUNITIES

This section discusses the previous challenges from the systems perspective. In particular, we discuss how innovations in systems, security, and architectures can help address these challenges. We present nine research opportunities (from **R1** to **R9**), organized into three topics: acting in dynamic environments, secure AI, and AI-specific architectures. Figure 1 shows the most common relationships between trends, on one hand, and challenges and research topics, on the other hand.

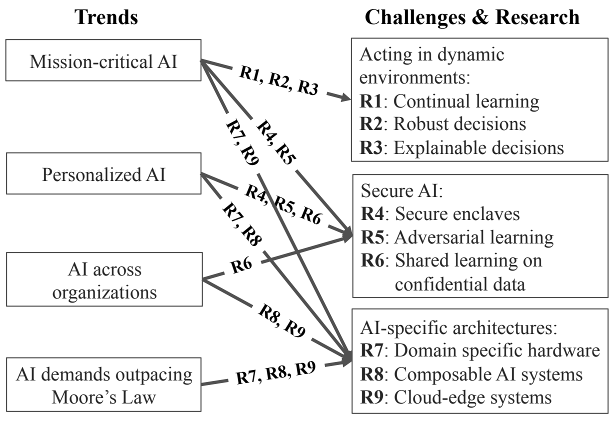

Figure 1: A mapping from trends to challenges and research topics.

4.1 Acting in dynamic environments

Many future AI applications will operate in *dynamic environments*, i.e., environments that may change, often rapidly and unexpectedly, and often in non-reproducible ways. For example, consider a group of robots providing security for an office building. When one robot breaks or a new one is added, the other robots must update their strategies for navigation, planning, and control in a coordinated manner. Similarly, when the environment changes, either due to the robots' own actions or to external conditions (e.g., an elevator going out of service, or a malicious intruder), all robots must re-calibrate their actions in light of the change. Handling such environments will require AI systems that can react quickly and safely even to scenarios that have not been encountered before.

R1: Continual learning. Most of today's AI systems, including movie recommendation, image recognition, and language translation, perform training offline and then make predictions online. That is, the learning performed by the system does not happen continually with the generation of the data, but instead it happens sporadicallly, on very different and much slower time scales. Typically, models are updated daily, or in the best case hourly, while predictions/decisions happen at second or sub-second granularity. This makes them a poor fit for environments that change continually and unexpectedly, especially in mission-critical applications. These more challenging environments require agents that *continually learn and adapt* to asynchronous changes.

Some aspects of learning in dynamic environments are addressed by online learning [17], in which data arrive temporally and updates to the model can occur as new data arrive. However, traditional online learning does not aim to handle control problems, in which an agent's actions change the environment (e.g., as arise naturally in robotics), nor does it aim to handle cases in which the outcomes of decisions are delayed (e.g., a move in a game of chess whose outcome is only evaluated at the *end*, when the game is lost or won).

These more general situations can be addressed in the framework of Reinforcement Learning (RL). The central task of RL is to learn a function—a "policy"—that maps observations (e.g., car's camera inputs or user's requested content) to actions (e.g., slowing down the car or presenting an ad) in a sequence that maximizes long-term reward (e.g., avoiding collisions or increasing sales). RL algorithms update the policy by taking into account the impact of agent's actions on the environment, even when delayed. If environmental changes lead to reward changes, RL updates the policy accordingly. RL has a long-standing tradition, with classical success stories including learning to play backgammon at level of the best human players [108], learning to walk [105], and learning basic motor skills [86]. However, these early efforts require significant tuning for each application. Recent efforts are combining deep neural networks with RL (Deep RL) to develop more robust training algorithms that can work for a variety of environments (e.g., many Atari games [77]), or even across different application domains, as in the control of (simulated) robots [92] and the learning of robotic manipulation skills [66]. Noteworthy recent results also include Google's AlphaGo beating the Go world champion [95], and new applications in medical diagnosis [104] and resource management [33].

However, despite these successes, RL has yet to see widescale real-world application. There are many reasons for this, one of which is that large-scale systems have not been built with these use cases in mind. We believe that the combination of ongoing advances in RL algorithms, when coupled with innovations in systems design, will catalyze rapid progress in RL and drive new RL applications.

Systems for RL. Many existing RL applications, such as game-playing, rely heavily on simulations, often requiring millions or even billions of simulations to explore the solution space and "solve" complex tasks. Examples include playing different variants of a game or experimenting with different control strategies in a robot simulator. These simulations can take as little as a few milliseconds, and their durations can be highly variable (e.g., it might take a few moves to lose a game vs. hundreds of moves to win one). Finally, real-world deployments of RL systems need to process inputs from a variety of sensors that observe the environment's state, and this must be accomplished under stringent time constraints. Thus, we need systems that can handle arbitrary dynamic task graphs, where tasks are heterogeneous in time, computation, and resource demands. Given the short duration of the simulations, to fully utilize a large cluster, we need to execute millions of simulations per second. None of the existing systems satisfies these requirements.

Data parallel systems [55, 79, 114] handle orders of magnitude fewer tasks per sec, while HPC and distributed DL systems [2, 23, 82] have limited support for heterogeneous and dynamic task graphs. Hence, we need new systems to support effectively RL applications.

Simulated reality (SR). The ability to interact with the environment is fundamental to RL's success. Unfortunately, in real-world applications, direct interaction can be slow (e.g., on the order of seconds) and/or hazardous (e.g., risking irreversible physical damage), both of which conflict with the need for having millions of interactions before a reasonable policy is learned. While algorithmic approaches have been proposed to reduce the number of real-world interactions needed to learn policies [99, 111, 112], more generally there is a need for *Simulated Reality (SR)* architectures, in which an agent can continually *simulate and predict* the outcome of the next action before actually taking it [101].

SR enables an agent to learn not only much faster but also much more safely. Consider a robot cleaning an environment that encounters an object it has not seen before, e.g., a new cellphone. The robot could physically experiment with the cellphone to determine how to grasp it, but this may require a long time and might damage the phone. In contrast, the robot could scan the 3D shape of the phone into a simulator, perform a few physical experiments to determine rigidity, texture, and weight distribution, and then use SR to learn how to successfully grasp it without damage.

Importantly, SR is quite different from virtual reality (VR); while VR focuses on simulating a hypothetical environment (e.g., Minecraft), sometimes incorporating past snapshots of the real world (e.g., Flight Simulator), SR focuses on continually simulating the physical world with which the agent is interacting. SR is also different from augmented reality (AR), which is primarily concerned with overlaying virtual objects onto real world images.

Arguably the biggest systems challenges associated with SR are to infer continually the simulator parameters in a changing real-world environment and at the same time to run many simulations before taking a single real-time action. As the learning algorithm interacts with the world, it gains more knowledge which can be used to improve the simulation. Meanwhile, many potential simulations would need to be run between the agent's actions, using both different potential plans and making different "what-if" assumptions about the world. Thus, the simulation is required to run much faster than real time.

Research: *(1) Build systems for RL that fully exploit parallelism, while allowing dynamic task graphs, providing millisecond-level latencies, and running on heterogeneous hardware under stringent deadlines. (2) Build systems that can faithfully simulate the real-world environment, as the environment changes continually and unexpectedly, and run faster than real time.*

R2: Robust decisions. As AI applications are increasingly making decisions on behalf of humans, notably in mission-critical applications, an important criterion is that they need to be robust to uncertainty and errors in inputs and feedback. While noise-resilient and robust learning is a core topic in statistics and machine learning, adding system support can significantly improve classical methods. In particular, by building systems that track data provenance, we can diminish uncertainty regarding the mapping of data sources to observations, as well as their impact on states and rewards. We can also track and leverage contextual information that informs the design of source-specific noise models (e.g., occluded cameras). These capabilities require support for provenance and noise modeling in data storage systems. While some of these challenges apply more generally, two notions of robustness that are particularly important in the context of AI systems and that present particular systems challenges are: (1) robust learning in the presence of noisy and adversarial feedback, and (2) robust decision-making in the presence of unforeseen and adversarial inputs.

Increasingly, learning systems leverage data collected from unreliable sources, possibly with inaccurate labels, and in some cases with deliberately inaccurate labels. For example, the Microsoft Tay chatbot relied heavily on human interaction to develop rich natural dialogue capabilities. However, when exposed to Twitter messages, Tay quickly took on a dark personality [16].

In addition to dealing with noisy feedback, another research challenge is handling inputs for which the system was never trained. In particular, one often wishes to detect whether a query input is drawn from a substantially different distribution than the training data, and then take safe actions in those cases. An example of a safe action in a self-driving car may be to slow down and stop. More generally, if there is a human in the loop, a decision system could relinquish control to a human operator. Explicitly training models to decline to make predictions for which they are not confident, or to adopt a default safe course of actions, and building systems that chain such models together can both reduce computational overhead and deliver more accurate and reliable predictions.

Research*: (1) Build fine grained provenance support into AI systems to connect outcome changes (e.g., reward or state) to the data sources that caused these changes, and automatically learn causal, source-specific noise models. (2) Design API and language support for developing systems that maintain confidence intervals for decision-making, and in particular can flag unforeseen inputs.*

R3: Explainable decisions. In addition to making black-box predictions and decisions, AI systems will often need to provide explanations for their decisions that are meaningful to humans. This is especially important for applications in which there are substantial regulatory requirements as well as in applications such as security and healthcare where legal issues arise [24]. Here, explainable should be distinguished from interpretable, which is often also of interest. Typically, the latter means that the output of the AI algorithm is understandable to a subject matter expert in terms of concepts from the domain from which the data are drawn [69], while the former means that one can identify the properties of the input to the AI algorithm that are responsible for the particular output, and can answer counterfactual or "what-if" questions. For example, one may wish to know what features of a particular organ in an X-ray (e.g., size, color, position, form) led to a particular diagnosis and how the diagnosis would change under minor perturbations of those features. Relatedly, one may wish to explore what other mechanisms could have led to the same outcomes, and the relative plausibility of those outcomes. Often this will require not merely providing an explanation for a decision, but also considering other data that could be brought to bear. Here we are in the domain of causal inference, a field which will be essential in many future AI applications, and one which has natural connections to diagnostics and provenance ideas in databases.

Indeed, one ingredient for supporting explainable decisions is the ability to *record and faithfully replay* the computations that led to a particular decision. Such systems hold the potential to help improve decision explainability by replaying a prediction task against past inputs—or randomly or adversarially perturbed versions of past inputs, or more general counterfactual scenarios—to identify what features of the input have caused a particular decision. For example, to identify the cause of a false alarm in a video-based security system, one might introduce perturbations in the input video that attenuate the alarm signal (e.g., by masking regions of the image) or search for closely related historical data (e.g., by identifying related inputs) that led to similar decisions. Such systems could also lead to improved statistical diagnostics and improved training/testing for new models; e.g., by designing models that are (or are not) amenable to explainability.

Research: *Build AI systems that can support interactive diagnostic analysis, that faithfully replay past executions, and that can help to determine the features of the input that are responsible for a particular decision, possibly by replaying the decision task against past perturbed inputs. More generally, provide systems support for causal inference.*

4.2 Secure AI

Security is a large topic, many aspects of which will be central to AI applications going forward. For example, mission-critical AI applications, personalized learning, and learning across multiple organizations all require systems with strong security properties. While there is a wide range of security issues, here we focus on two broad categories. The first category is an attacker compromising the integrity of the decision process. The attacker can do so either by compromising and taking the control of the AI system itself, or by altering the inputs so that the system will unknowingly render decisions that the attacker wants. The second category is an attacker learning the confidential data on which an AI system was trained on, or learning the secret model. Next, we discuss three promising research topics to defend against such attacks.

R4: Secure enclaves. The rapid rise of public cloud and the increased complexity of the software stack considerably widen the exposure of AI applications to attacks. Two decades ago most applications ran on top of a commercial OS, such as Windows or SunOS, on a single server deployed behind organization's firewalls. Today, organizations run AI applications in the public cloud on a distributed set of servers they do not control, possibly shared with competitors, on a considerably more complex software stack, where the OS itself runs on top of a hypervisor or within a container. Furthermore, the applications leverage directly or indirectly a plethora of other systems, such as log ingestion, storage, and data processing frameworks. If any of these software components is compromised, the AI applications itself might be compromised.

A general approach to deal with these attacks is providing a "secure enclave" abstraction—a secure execution environment—which protects the application running within the enclave from malicious code running outside the enclave. One recent example is Intel's Software Guard Extensions (SGX) [5], which provides a hardware-enforced isolated execution environment. Code inside SGX can compute on data, while even a compromised operating system or hypervisor (running outside the enclave) cannot see this code or data. SGX also provides remote attestation [6], a protocol enabling a remote client to verify that the enclave is running the expected code. ARM's TrustZone is another example of a hardware enclave. At the other end of the spectrum, cloud providers are starting to offer special bare-bone instances that are physically protected, e.g., they are deployed in secure "vaults" to which only authorized personnel, authenticated via fingerprint or iris scanning, has access.

In general, with any enclave technology, the application developer must trust all the software running within the enclave. Indeed, even in the case of hardware enclaves, if the code running inside the enclave is compromised, it can leak decrypted data or compromise decisions. Since a small code base is typically easier to secure, one research challenge is to split the AI system's code into code running inside the enclave, hopefully as little as possible, and code running outside of the enclave, in untrusted mode, by leveraging cryptographic techniques. Another approach to ensure that code inside the enclave does not leak sensitive information is to develop static and dynamic verification tools as well as sandboxing [9, 12, 93].

Note that beside minimizing the trusted computing base, there are two additional reasons for splitting the application code: increased functionality and reduced cost. First, some of the functionality might not be available within the enclave, e.g., GPU processing for running Deep Learning (DL) algorithms, or services and applications which are not vetted/ported yet to run within the secure enclave. Second, the secure instances offered by a cloud provider can be significantly more expensive than regular instances.

Research: *Build AI systems that leverage secure enclaves to ensure data confidentiality, user privacy and decision integrity, possibly by splitting the AI system's code between a minimal code base running within the enclave, and code running outside the enclave. Ensure the code inside the enclave does not leak information, or compromise decision integrity.*

R5: Adversarial learning. The adaptive nature of ML algorithms opens the learning systems to new categories of attacks that aim to compromise the integrity of their decisions by maliciously altering training data or decision input. There are two broad types of attacks: *evasion attacks* and *data poisoning attacks*.

Evasion attacks happen at the inference stage, where an adversary attempts to craft data that is incorrectly classified by the learning system [47, 103]. An example is altering the image of a stop sign slightly such that it still appears to a human to be a stop sign but is seen by an autonomous vehicle as a yield sign.

Data poisoning attacks happen at the training stage, where an adversary injects poisoned data (e.g., data with wrong labels) into the training data set that cause the learning system to learn the wrong model, such that the adversary thereby has input data incorrectly classified by the learner [73, 74, 113]. Learning systems that are periodically retrained to handle non-stationary input data are particularly vulnerable to this attack, if the weakly labeled data being used for retraining is collected from unreliable or untrustworthy sources. With new AI systems continually learning by interacting with dynamic environments, handling data poisoning attacks becomes increasingly important.

Today, there are no effective solutions to protect against evasion attacks. As such, there are a number of open research challenges: provide better understanding of why adversarial examples are often easy to find, investigate what method or combination of different methods may be effective at defending against adversarial examples, and design and develop systematic methods to evaluate potential defenses. For data poisoning attacks, open challenges include how to detect poisoned input data and how to build learning systems that are resilient to different types of data poisoning attacks. In addition, as data sources are identified to be fraudulent or explicitly retracted for regulatory reasons, we can leverage replay (see R3: Explainable decisions) and incremental computation to efficiently eliminate the impact of those sources on learned models. As pointed out previously, this ability is enabled by combining modeling with provenance and efficient computation in data storage systems.

Research: *Build AI systems that are robust against adversarial inputs both during training and prediction (e.g., decision making), possibly by designing new machine learning models and network architectures, leveraging provenance to track down fraudulent data sources, and replaying to redo decisions after eliminating the fraudulent sources.*

R6: Shared learning on confidential data. Today, each company typically collects data individually, analyzes it, and uses this data to implement new features and products. However, not all organizations possess the same wealth of data as found in the few large AI-focused corporations, such as Google, Facebook, Microsoft, and Amazon. Going forward, we expect more and more organizations to collect valuable data, more third-party data services to be available, and more benefit to be gained from learning over data from multiple organizations (see Section 3).

Indeed, from our own interaction with industry, we are learning about an increasing number of such scenarios. A large bank provided us with a scenario in which they and other banks would like to pool together their data and use shared learning to improve their collective fraud detection algorithms. While these banks are natural competitors in providing financial services, such "cooperation" is critical to minimize their losses due to fraudulent activities. A very large health provider described a similar scenario in which competing hospitals would like to share data to train a shared model predicting flu outbreaks without sharing the data for other purposes. This would allow them to improve the response to epidemics and contain the outbreaks, e.g., by rapidly deploying mobile vaccination vans at critical locations. At the same time, every hospital must protect the confidentiality of their own patients.

The key challenge of shared learning is how to learn a model on data belonging to different (possible competitive) organizations without leaking relevant information about this data during the training process. One possible solution would be to pool all the data in a hardware enclave and then learn the model. However, this solution is not always feasible as hardware enclaves are not yet deployed widely, and, in some cases, the data cannot be moved due to regulatory constraints or its large volume.

Another promising approach is using *secure multi-party computation (MPC)* [13, 45, 70]. MPC enables *n* parties, each having a private input, to compute a joint function over the input without any party learning the inputs of the other parties. Unfortunately, while MPC is effective for simple computations, it has a nontrivial overhead for complex computations, such as model training. An interesting research direction is how to partition model training into (1) local computation and (2) computation using MPC, so that we minimize the complexity of the MPC computation.

While training a model without compromising data confidentiality is a big step towards enabling shared learning, unfortunately, it is not always enough. Model serving—the inferences (decisions) rendered based on the model—can still leak information about the data [42, 94]. One approach to address this challenge is *differential privacy* [36, 37, 39], a popular technique proposed in the context of statistical databases. Differential privacy adds noise to each query to protect the data privacy, hence effectively trading accuracy for privacy [35]. A central concept of differential privacy is the privacy budget which caps the number of queries given a privacy guarantee.

There are three interesting research directions when applying differential privacy to model serving. First, a promising approach is to leverage differential privacy for complex models and inferences, by taking advantage of the inherent statistical nature of the models and predictions. Second, despite the large volume of theoretical research, there are few practical differential privacy systems in use today. An important research direction is to build tools and systems to make it easy to enable differential privacy for real-world applications, including intelligently selecting which privacy mechanisms to use for a given application and automatically converting non-differentially-private computations to differentially-private computations. Finally, one particular aspect in the context of continual learning is that data privacy can be time dependent, that is, the privacy of fresh data is far more important than the privacy of historical data. Examples are stock market and online bidding, where the privacy of the fresh data is paramount, while the historical data is sometimes publicly released. This aspect could enable the development of new differential privacy systems with adaptive privacy budgets that apply only to decisions on the most recent data. Another research direction is to further develop the notion of differential privacy under continuous observation and data release [21, 38].

Even if we are able to protect data confidentiality during training and decision making, this might still not be enough. Indeed, even if confidentiality is guaranteed, an organization might refuse to share its data for improving a model from which its competitors might benefit. Thus, we need to go beyond guaranteeing confidentiality and provide *incentives* to organizations to share their data or byproducts of their data. Specifically, we need to develop approaches that ensure that by sharing data, an organization gets strictly better service (i.e., better decisions) than not sharing data. This requires ascertaining the quality of the data providing by a given organization—a problem which can be tackled via a "leave-one-out" approach in which performance is compared both with and without that organization's data included in the training set. We then provide decisions that are corrupted by noise at a level that is inversely proportional to the quality of the data provided by an organization. This incentivizes an organization to provide higher-quality data. Overall, such incentives will need to be placed within a framework of mechanism design to allow organizations to forge their individual data-sharing strategies.

Research: *Build AI systems that (1) can learn across multiple data sources without leaking information from a data source during training or serving, and (2) provide incentives to potentially competing organizations to share their data or models.*

4.3 AI-specific architectures

AI demands will drive innovations both in systems and hardware architectures. These new architectures will aim not only to improve the performance, but to simplify the development of the next generation of AI applications by providing rich libraries of modules that are easily composable.

R7: Domain specific hardware. The ability to process and store huge amounts of data has been one of the key enablers of the AI's recent successes (see Section 2.1). However, continuing to keep up with the data being generated will be increasingly challenging. As discussed in Section 3, while data continues to grow exponentially, the corresponding performance-cost-energy improvements that have fueled the computer industry for more than 40 years are reaching the end-of-line:

- Transistors are not getting much smaller due to the ending of Moore's Law,
- Power is limiting what can be put on a chip due to the end of Dennard scaling,
- We've already switched from one inefficient processor/chip to about a dozen efficient processors per chip, but there are limits to parallelism due to Amdahl's Law.

The one path left to continue the improvements in performance-energy-cost of processors is developing domain-specific processors. These processors do only a few tasks, but they do them extremely well. Thus, the rapid improvements in processing that we have expected in the Moore's law era must now come through innovations in computer architecture instead of semiconductor process improvements. Future servers will have much more heterogeneous processors than in the past. One trailblazing example that spotlights domain specific processors is Google's Tensor Processing Unit, which has been deployed in its datacenters since 2015 and is regularly used by billions of people. It performs the inference phase of deep neural networks 15× to 30× faster than its contemporary CPUs and GPUs and its performance per watt is 30× to 80× better. In addition, Microsoft has announced the availability of FPGA-powered instances on its Azure cloud [88], and a host of companies, ranging from Intel to IBM, and to startups like Cerebras and Graphcore are developing specialized hardware for AI that promise orders of magnitude performance improvements over today's state-of-the-art processors [19, 48, 54, 78].

With DRAM subject to the same limitations, there are several novel technologies being developed that hope to be its successor. 3D XPoint from Intel and Micron aims to provide 10× storage capacity with DRAM-like performance. STT MRAM aims to succeed Flash, which may hit similar scaling limits as DRAM. Hence, the memory and storage of the cloud will likely have more levels in the hierarchy and contain a wider variety of technologies. Given the increasing diversity of processors, memories, and storage devices, mapping services to hardware resources will become an even more challenging problem. These dramatic changes suggest building cloud computing from a much more flexible building block than the classic standard rack containing a top-of-rack switch and tens of servers, each with 2 CPU chips, 1 TB of DRAM, and 4 TBs of flash.

For example, the UC Berkeley Firebox project [41] proposes a multi-rack supercomputer that connects thousands of processor chips with thousands of DRAM chips and nonvolatile storage chips using fiber optics to provide low-latency, high-bandwidth, and long physical distance. Such a hardware system would allow system software to provision computation services with the right ratio and type of domain-specific processors, DRAM, and NVRAM. Such resource disaggregation at scale would significantly improve the allocation of increasingly diverse tasks to correspondingly heterogeneous resources. It is particularly valuable for AI workloads, which are known to gain significant performance benefits from large memory and have diverse resource requirements that don't all conform to a common pattern.

Besides performance improvements, new hardware architectures will also provide additional functionality, such as security support. While Intel's SGX and ARM's TrustZone are paving the way towards hardware enclaves, much more needs to be done before they can be fully embraced by AI applications. In particular, existing enclaves exhibit various resource limitations such as addressable memory, and they are only available for a few general purpose CPUs. Removing these limitations, and providing a uniform hardware enclave abstraction across a diverse set of specialized processors, including GPUs and TPUs, are promising directions of research. In addition, open instruction set processors, such as RISC-V represent an exciting "playground" to develop new security features.

Research: *(1) Design domain-specific hardware architectures to improve the performance and reduce power consumption of AI applications by orders of magnitude, or enhance the security of these applications. (2) Design AI software systems to take advantage of these domain-specific architectures, resource disaggregation architectures, and future non-volatile storage technologies.*

R8: Composable AI systems. Modularity and composition have played a fundamental role in the rapid progress of software systems, as they allowed developers to rapidly build and evolve new systems from existing components. Examples range from microkernel OSes [3, 68], LAMP stack [64], microservice architectures [85], and the internet [26]. In contrast, today's AI systems are monolithic which makes them hard to develop, test, and evolve.

Similarly, modularity and composition will be key to increasing development speed and adoption of AI, by making it easier to integrate AI in complex systems. Next, we discuss several research problems in the context of model and action composition.

Model composition is critical to the development of more flexible and powerful AI systems. Composing multiple models and querying them in different patterns enables a tradeoff between decision accuracy, latency, and throughput in a model serving system [29, 106] In one example, we can query models serially, where each model either renders the decision with sufficiently high accuracy or says "I'm not sure". In the latter case, the decision is passed to the next model in the series. By ordering the models from the highest to the lowest "I'm not sure" rate, and from lowest to the highest latency, we can optimize both latency and accuracy.

To fully enable model composition, many challenges remain to be addressed. Examples are (1) designing a declarative language to capture the topology of these components and specifying performance targets of the applications, (2) providing accurate performance models for each component, including resource demands, latency and throughput, and (3) scheduling and optimization algorithms to compute the execution plan across components, and map components to the available resources to satisfy latency and throughput requirements while minimizing costs.

Action composition consists of aggregating sequences of basic decisions/actions into coarse-grained primitives, also called *options*. In the case of a self-driving car, an example of an option is changing the lane while driving on a highway, while the actions are speeding up, slowing down, turning left or right, signaling the change of direction, etc. In the case of a robot, an example of a primitive could be grasping an object, while actions include actuating the robot's joints. Options have been extensively studied in the context of hierarchical learning [30, 34, 84, 97, 102, 110]. Options can dramatically speed up learning or adaptation to a new scenario by allowing the agent to select from a list of existing options to accomplish a given task, rather than from a much longer list of low-level actions.

A rich library of options would enable the development of new AI applications by simply composing the appropriate options the same way web programmers develop applications today in just a few lines of code by invoking powerful web APIs. In addition, options can improve responsiveness as selecting the next action within an option is a much simpler task than selecting an action in the original action space.

Research: *Design AI systems and APIs that allow the composition of models and actions in a modular and flexible manner, and develop rich libraries of models and options using these APIs to dramatically simplify the development of AI applications.*

R9: Cloud-edge systems. Today, many AI applications such as speech recognition and language translation are deployed in the cloud. Going forward we expect a rapid increase in AI systems that span edge devices and the cloud. On one hand, AI systems which are currently cloud only, such as user recommendation systems [72], are moving some of their functionality to edge devices to improve security, privacy, latency and safety (including the ability to cope with being disconnected from the internet). On the other hand, AI systems currently deployed at the edge, such as self-driving cars, drones, and home robots, are increasingly sharing data and leveraging the computational resources available in the cloud to update models and policies [61].

However, developing cloud and the cloud-edge systems is challenging for several reasons. First, there is a large discrepancy between the capabilities of edge devices and datacenter servers. We expect this discrepancy to increase in the future, as edge devices, such as cellphones and tablets, have much more stringent power and size constraints than servers in datacenters. Second, edge devices are extremely heterogeneous both in terms of resource capabilities, ranging from very low power ARM or RISC-V CPUs that power IoT devices to powerful GPUs in self-driving cars, and software platforms. This heterogeneity makes application development much harder. Third, the hardware and software update cycles of edge devices are significantly slower than in a datacenter. Fourth, as the increase in the storage capacity slows down, while the growth in the data being generated continues unabated, it may no longer be feasible or cost effective to store this deluge of data.

There are two general approaches to addressing the mix of cloud and edge devices. The first is to repurpose code to multiple heterogeneous platforms via retargetable software design and compiler technology. To address the wide heterogeneity of edge devices and the relative difficulty of upgrading the applications running on these devices, we need new software stacks that abstract away the heterogeneity of devices by exposing the hardware capabilities to the application through common APIs. Another promising direction is developing compilers and just-in-time (JIT) technologies to efficiently compile on-the-fly complex algorithms and run them on edge devices. This approach can leverage recent code generation tools, such as TensorFlow's XLA [107], Halide [50], and Weld [83].

The second general approach is to design AI systems that are well suited to partitioned execution across the cloud and the edge. As one example, model composition (see Section 4.3) could allow one to run the lighter but less accurate models at the edge, and the computation-intensive but higher-accuracy models in the cloud. This architecture would improve decision latency, without compromising accuracy, and it has been already employed in recent video recognition systems [59, 115]. In another example, action composition would allow building systems where learning of hierarchical options [63] takes place on powerful clusters in the cloud, and then execution of these options happens at the edge.

Robotics is one application domain that can take advantage of a modular cloud-edge architecture. Today, there is a scarcity of open source platforms to develop robotic applications. ROS, arguably the most popular such platform in use today, is confined to running locally and lacks many performance optimizations required by real-time applications. To take advantage of the new developments in AI research such as shared and continual learning, we need systems that can span both edge devices (e.g., robots) and the cloud. Such systems would allow developers to seamlessly migrate the functionality between a robot and the cloud to optimize decision latency and learning convergence. While the cloud can run sophisticated algorithms to continually update the models by leveraging the information gathered by distributed robots in real time, the robots can continue to execute the actions locally based on previously downloaded policies.

To address the challenge of the data deluge collected by the edge devices, learning-friendly compression methods can be used to reduce processing overhead. Examples of such methods include sampling and sketching, which have already been successfully employed for analytics workloads [4, 10, 28, 51, 81]. One research direction is to aggressively leverage sampling and sketching in a systematic way to support a variety of learning algorithms and prediction scenarios. An arguably more difficult challenge is to reduce the storage overhead, which might require to delete data. The key challenge here is that we do not always know how the data will be used in the future. This is essentially a compression problem, but compression for the purposes of ML algorithms. Again, distributed approaches based in materialized samples and sketches can help provide solutions to this problem, as can ML-based approaches in the form of feature selection or model selection protocols.

Research: *Design cloud-edge AI systems that (1) leverage the edge to reduce latency, improve safety and security, and implement intelligent data retention techniques, and (2) leverage the cloud to share data and models across edge devices, train sophisticated computation-intensive models, and take high quality decisions.*

5 CONCLUSION

The striking progress of AI during just the last decade is leading to the successful transition from the research lab into commercial services that have previously required human input and oversight. Rather than replacing human workers, AI systems and robots have potential to enhance human performance and facilitate new forms of collaboration [44].

To realize the full promise of AI as a positive force in our lives, there are daunting challenges to overcome, and many of these challenges are related to systems and infrastructure. These challenges are driven by the realization that AI systems will need to make decisions that are faster, safer, and more explainable, securing these decisions as well as the learning processes against ever more sophisticated types of attacks, continuously increasing the computation capabilities in the face of the end of Moore's Law, and building composable systems that are easy to integrate in existing applications and can span the cloud and the edge.

This paper proposes several open research directions in systems, architectures, and security that have potential to address these challenges. We hope these questions will inspire new research that can advance AI and make it more capable, understandable, secure and reliable.

REFERENCES

[1] A History of Storage Cost. 2017. http://www.mkomo.com/cost-per-gigabyte-update. (2017).

[2] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, and Matthieu Devin. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. (2015).

[3] Mike Accetta, Robert Baron, William Bolosky, David Golub, Richard Rashid, Avadis Tevanian, and Michael Young. 1986. Mach: A New Kernel Foundation for UNIX Development. 93–112.

[4] Sameer Agarwal et al. 2013. BlinkDB: queries with bounded errors and bounded response times on very large data. In *EuroSys*.

[5] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. 2013. Innovative technology for CPU based attestation and sealing. In *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, Vol. 13.

[6] Ittai Anati, Shay Gueron, Simon P Johnson, and Vincent R Scarlata. 2013. Innovative Technology for CPU Based Attestation and Sealing. (2013).

[7] Apache Hadoop. 2017. http://hadoop.apache.org/. (2017).

[8] Apache Mahout. 2017. http://mahout.apache.org/. (2017).

[9] Sergei Arnautov, Bohdan Trach, Franz Gregor, Thomas Knauth, Andre Martin, Christian Priebe, Joshua Lind, Divya Muthukumaran, Daniel OâĂŹKeeffe, Mark L Stillwell, et al. 2016. SCONE: Secure linux containers with Intel SGX. In *12th USENIX Symp. Operating Systems Design and Implementation.*

[10] Peter Bailis, Edward Gan, Samuel Madden, Deepak Narayanan, Kexin Rong, and Sahaana Suri. 2017. MacroBase: Prioritizing Attention in Fast Data. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17).* ACM, New York, NY, USA, 541–556.

[11] Luiz Andre Barroso and Urs Hoelzle. 2009. *The Datacenter As a Computer: An Introduction to the Design of Warehouse-Scale Machines.* Morgan and Claypool.

[12] Andrew Baumann, Marcus Peinado, and Galen Hunt. 2015. Shielding applications from an untrusted cloud with haven. *ACM Transactions on Computer Systems (TOCS)* 33, 3 (2015), 8.

[13] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. 1988. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the 20th ACM symposium on Theory of Computing.*

[14] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, Vol. 4. Austin, TX, 3.

[15] bigdl. BigDL: Distributed Deep Learning on Apache Spark. https://software.intel.com/en-us/articles/bigdl-distributed-deep-learning-on-apache-spark. (????).

[16] Tay (bot). 2017. https://en.wikipedia.org/wiki/Tay_(bot). (2017).

[17] Léon Bottou. 1998. On-line Learning in Neural Networks. (1998), 9–42.

[18] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010.* Springer, 177–186.

[19] Cerebras. 2017. https://www.cerebras.net/. (2017).

[20] Chainer. 2017. https://chainer.org/. (2017).

[21] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. 2010. Private and Continual Release of Statistics. In *ICALP (2)*, Vol. 6199. Springer.

[22] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. *arXiv preprint arXiv:1512.01274* (2015).

[23] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. *CoRR* abs/1512.01274 (2015).

[24] Travers Ching, Daniel S Himmelstein, Brett K Beaulieu-Jones, Alexandr A Kalinin, Brian T Do, Gregory P Way, Enrico Ferrero, Paul-Michael Agapow, Wei Xie, Gail L Rosen, et al. 2017. Opportunities And Obstacles For Deep Learning In Biology And Medicine. *bioRxiv* (2017), 142760.

[25] cisco. 2015. Cisco Global Cloud Index: Forecast and Methodology, 2015-2020. http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf. (2015).

[26] D. Clark. 1988. The Design Philosophy of the DARPA Internet Protocols. *SIGCOMM Comput. Commun. Rev.* 18, 4 (Aug. 1988), 106–114.

[27] CMS updates rule allowing claims data to be sold. 2016. http://www.modernhealthcare.com/article/20160701/NEWS/160709998. (2016).

[28] Graham Cormode, Minos Garofalakis, Peter J Haas, and Chris Jermaine. 2012. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases* 4, 1–3 (2012), 1–294.

[29] Daniel Crankshaw, Xin Wang, Giulio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. 2017. Clipper: A Low-Latency Online Prediction Serving System. *NSDI '17* (2017).

[30] Peter Dayan and Geoffrey E. Hinton. 1992. Feudal Reinforcement Learning. In *Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992].* 271–278. http://papers.nips.cc/paper/714-feudal-reinforcement-learning

[31] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marcaurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc Le, and Andrew Y. Ng. 2012. Large Scale Distributed Deep Networks. In *NIPS '12.* http://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks.pdf

[32] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6 (OSDI'04).*

[33] DeepMind AI Reduces Google Data Centre Cooling Bill by 40%. 2017. https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/. (2017).

[34] Thomas G. Dietterich. 1998. The MAXQ Method for Hierarchical Reinforcement Learning. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998.* 118–126.

[35] John Duchi, Michael Jordan, and Martin Wainwright. to appear. Minimax optimal procedures for locally private estimation. *J. Amer. Statist. Assoc.* (to appear).

[36] Cynthia Dwork. 2006. Differential Privacy. In *ICALP (2)*, Vol. 4052. Springer.

[37] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation.*

[38] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. 2010. Differential privacy under continual observation. In *Proceedings of the 42nd ACM symposium on Theory of computing.*

[39] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science* 9 (2014).

[40] The Economist. 2017. The world's most valuable resource is no longer oil, but data. (May 2017).

[41] FireBox. 2017. https://bar.eecs.berkeley.edu/projects/2015-firebox.html. (2017).

[42] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security.* ACM, 1322–1333.

[43] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google File System. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (SOSP '03).* 29–43.

[44] Ken Goldberg. 2017. Op-Ed: Call it Multiplicity: Diverse Groups of People and Machines Working Together. *Wall Street Journal* (2017).

[45] Oded Goldreich, Silvio Micali, and Avi Wigderson. 1987. How to play any mental game. In *Proceedings of the 19th ACM symposium on Theory of computing.*

[46] Joseph E. Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin. 2012. PowerGraph: Distributed Graph-parallel Computation on Natural Graphs *(OSDI'12).* 17–30.

[47] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[48] Graohcore. 2017. https://www.graphcore.ai/. (2017).

[49] Alon Halevy, Peter Norvig, , and Fernando Pereira. 2009. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems* 24, 2 (2009), 8–12.

[50] Halide: A Language for Image Processing and Computational Photography. 2017. http://halide-lang.org/. (2017).

[51] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. 1997. Online Aggregation. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data (SIGMOD '97)*. ACM, New York, NY, USA, 171–182. https://doi.org/10.1145/253260.253291

[52] Joseph M. Hellerstein, Christoper Ré, Florian Schoppmann, Daisy Zhe Wang, Eugene Fratkin, Aleksander Gorajek, Kee Siong Ng, Caleb Welton, Xixuan Feng, Kun Li, and Arun Kumar. 2012. The MADlib Analytics Library: Or MAD Skills, the SQL. *Proc. VLDB Endow.* 5, 12 (Aug. 2012), 1700–1711.

[53] John L. Hennessy and David A. Patterson. to appear. Computer Architecture, Sixth Edition: A Quantitative Approach. (to appear).

[54] Intel Nervana. 2017. https://www.intelnervana.com/intel-nervana-hardware/. (2017).

[55] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. 2007. Dryad: Distributed Data-parallel Programs from Sequential Building Blocks. In *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007 (EuroSys '07)*. 59–72.

[56] Martin Jaggi, Virginia Smith, Martin Takac, Jonathan Terhorst, Sanjay Krishnan, Thomas Hoffmann, and Michael I. Jordan. 2015. Communication-Efficient Distributed Dual Coordinate Ascent. In *NIPS, 27*.

[57] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*. ACM, 675–678.

[58] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA '17)*. ACM, New York, NY, USA, 1–12. https://doi.org/10.1145/3079856.3080246

[59] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. Optimizing Deep CNN-Based Queries over Video Streams at Scale. *CoRR* abs/1703.02529 (2017).

[60] Asterios Katsifodimos and Sebastian Schelter. 2016. Apache Flink: Stream Analytics at Scale.

[61] Ben Kehoe, Sachin Patil, Pieter Abbeel, and Ken Goldberg. 2015. A Survey of Research on Cloud Robotics and Automation. *IEEE Trans. Automation Science and Eng.* 12, 2 (2015).

[62] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). http://arxiv.org/abs/1412.6980

[63] Sanjay Krishnan, Roy Fox, Ion Stoica, and Ken Goldberg. 2017. DDCO: Discovery of Deep Continuous Options for Robot Learning from Demonstrations. In *1st Conference on Robot Learning (CoRL)*.

[64] LAMP (software bundle). 2017. https://en.wikipedia.org/wiki/LAMP_(software_bundle). (2017).

[65] Leo Leung. 2015. How much data does x store? (March 2015). https://techexpectations.org/tag/how-much-data-does-youtube-store/

[66] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end Training of Deep Visuomotor Policies. *J. Mach. Learn. Res.* 17, 1 (Jan. 2016), 1334–1373. http://dl.acm.org/citation.cfm?id=2946645.2946684

[67] 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 *OSDI '14*. 583–598.

[68] J. Liedtke. 1995. On Micro-kernel Construction. In *Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles (SOSP '95)*. ACM, New York, NY, USA, 237–250. https://doi.org/10.1145/224056.224075

[69] M. W. Mahoney and P. Drineas. 2009. CUR Matrix Decompositions for Improved Data Analysis. *Proc. Natl. Acad. Sci. USA* 106 (2009), 697–702.

[70] Dahlia Malkhi, Noam Nisan, Benny Pinkas, Yaron Sella, et al. 2004. Fairplay-Secure Two-Party Computation System.. In *USENIX Security Symposium*, Vol. 4. San Diego, CA, USA.

[71] Michael Mccloskey and Neil J. Cohen. 1989. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *The Psychology of Learning and Motivation* 24 (1989), 104–169.

[72] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*. http://arxiv.org/abs/1602.05629

[73] Shike Mei and Xiaojin Zhu. 2015. The Security of Latent Dirichlet Allocation.. In *AISTATS*.

[74] Shike Mei and Xiaojin Zhu. 2015. Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners.. In *AAAI*. 2871–2877.

[75] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Zadeh, Matei Zaharia, and Ameet Talwalkar. 2016. MLlib: Machine Learning in Apache Spark. *Journal of Machine Learning Research* 17, 34 (2016), 1–7. http://jmlr.org/papers/v17/15-237.html

[76] Tom M Mitchell, William W Cohen, Estevam R Hruschka Jr, Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, et al. 2015. Never Ending Learning.. In *AAAI*. 2302–2310.

[77] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (26 02 2015), 529–533. http://dx.doi.org/10.1038/nature14236

[78] Dharmendra Modha. 2016. The brainâĂŹs architecture, efficiencyâĂę on a chip. (Dec. 2016). https://www.ibm.com/blogs/research/2016/12/the-brains-architecture-efficiency-on-a-chip/

[79] Derek G. Murray, Malte Schwarzkopf, Christopher Smowton, Steven Smith, Anil Madhavapeddy, and Steven Hand. 2011. CIEL: A Universal Execution Engine for Distributed Data-flow Computing. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI'11)*. USENIX Association, Berkeley, CA, USA, 113–126. http://dl.acm.org/citation.cfm?id=1972457.1972470

[80] Average Historic Price of RAM. 2017. http://www.statisticbrain.com/average-historic-price-of-ram/. (2017).

[81] Frank Olken and Doron Rotem. 1990. Random sampling from database files: A survey. *Statistical and Scientific Database Management* (1990), 92–111.

[82] Open MPI: Open Source High Performance Computing. 2017. https://www.open-mpi.org/. (2017).

[83] Shoumik Palkar, James J. Thomas, Anil Shanbhag, Deepak Narayanan, Holger Pirk, Malte Schwarzkopf, Saman Amarasinghe, and Matei Zaharia. 2017. Weld: A Common Runtime for High Performance Data Analytics. In *CIDR*.

[84] Ronald Parr and Stuart J. Russell. 1997. Reinforcement Learning with Hierarchies of Machines. In *Advances in Neural Information Processing Systems 10, [NIPS Conference, Denver, Colorado, USA, 1997]*. 1043–1049. http://papers.nips.cc/paper/1384-reinforcement-learning-with-hierarchies-of-machines

[85] Pattern: Microservice Architecture. 2017. http://microservices.io/patterns/microservices.html. (2017).

[86] Jan Peters and Stefan Schaal. 2008. Reinforcement learning of motor skills with policy gradients. *Neural networks* 21, 4 (2008), 682–697.

[87] Gil Press. 2016. Forrester Predicts Investment In Artificial Intelligence Will Grow 300% in 2017. *Forbes* (November 2016).

[88] Project Catapult. 2017. https://www.microsoft.com/en-us/research/project/project-catapult/. (2017).

[89] PyTorch. 2017. http://pytorch.org/. (2017).

[90] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. 2009. Large-scale Deep Unsupervised Learning Using Graphics Processors. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*. ACM, New York, NY, USA, 873–880. https://doi.org/10.1145/1553374.1553486

[91] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. In *NIPS 24*.

[92] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2015. Trust Region Policy Optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*.

[93] Felix Schuster, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. 2015. VC3: Trustworthy data analytics in the cloud using SGX. In *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 38–54.

[94] Reza Shokri, Marco Stronati, and Vitaly Shmatikov. 2016. Membership inference attacks against machine learning models. *arXiv preprint arXiv:1610.05820* (2016).

[95] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, and others. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[96] Daniel L Silver, Qiang Yang, and Lianghao Li. 2013. Lifelong Machine Learning Systems: Beyond Learning Algorithms.. In *AAAI Spring Symposium: Lifelong Machine Learning*, Vol. 13. 05.

[97] Satinder P. Singh. 1992. Reinforcement Learning with a Hierarchy of Abstract Models. In *Proceedings of the 10th National Conference on Artificial Intelligence.*

San Jose, CA, July 12-16, 1992. 202–207. http://www.aaai.org/Library/AAAI/1992/aaai92-032.php

[98] Evan R Sparks et al. 2013. MLI: An API for distributed machine learning. In *ICDM*.

[99] Stephane Ross. 2013. Interactive Learning for Sequential Decisions and Predictions. https://en.wikipedia.org/wiki/LAMP_(software_bundle). (2013).

[100] Zachary D Stephens, Skylar Y Lee, Faraz Faghri, Roy H Campbell, Chengxiang Zhai, Miles J Efron, Ravishankar Iyer, Michael C Schatz, Saurabh Sinha, and Gene E Robinson. 2015. Big data: Astronomical or genomical? *PLoS Biology* 13, 7 (2015), e1002195.

[101] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*. Morgan Kaufmann.

[102] Richard S. Sutton, Doina Precup, and Satinder P. Singh. 1999. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell.* 112, 1-2 (1999), 181–211. https://doi.org/10.1016/S0004-3702(99)00052-1

[103] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[104] Kai-Fu Tang, Hao-Cheng Kao, Chun-Nan Chou, and Edward Y. Chang. 2016. Inquire and Diagnose: Neural Symptom Checking Ensemble using Deep Reinforcement Learning. http://infolab.stanford.edu/~echang/NIPS_DeepRL_2016_Symptom_Checker.pdf. (2016).

[105] Russ Tedrake, Teresa Weirui Zhang, and H Sebastian Seung. 2005. Learning to walk in 20 minutes. In *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, Vol. 95585. Yale University New Haven (CT), 1939–1412.

[106] TensorFlow Serving. 2017. https://tensorflow.github.io/serving/. (2017).

[107] TensorFlow XLA. 2017. https://www.tensorflow.org/performance/xla/. (2017).

[108] Gerald Tesauro. 1995. Temporal difference learning and TD-Gammon. *Commun. ACM* 38, 3 (1995), 58–68.

[109] Sebastian Thrun. 1998. Lifelong learning algorithms. *Learning to learn* 8 (1998), 181–209.

[110] Sebastian Thrun and Anton Schwartz. 1994. Finding Structure in Reinforcement Learning. In *Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]*. 385–392. http://papers.nips.cc/paper/887-finding-structure-in-reinforcement-learning

[111] Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. *CoRR* abs/1703.06907 (2017). http://arxiv.org/abs/1703.06907

[112] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep Domain Confusion: Maximizing for Domain Invariance. *CoRR* abs/1412.3474 (2014). http://arxiv.org/abs/1412.3474

[113] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. 2015. Is feature selection secure against training data poisoning?. In *ICML*. 1689–1698.

[114] Matei Zaharia et al. 2012. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI '12*.

[115] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodík, Matthai Philipose, Paramvir Bahl, and Michael J. Freedman. 2017. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In *NSDI*.