Byzantine Machine Learning: A Primer

RACHID GUERRAOUI, NIRUPAM GUPTA, and RAFAEL PINOT, École polytechnique fédérale de Lausanne, Switzerland

The problem of Byzantine resilience in distributed machine learning, a.k.a. *Byzantine machine learning*, consists of designing distributed algorithms that can train an accurate model despite the presence of Byzantine nodes—that is, nodes with corrupt data or machines that can misbehave arbitrarily. By now, many solutions to this important problem have been proposed, most of which build upon the classical stochastic gradient descent scheme. Yet, the literature lacks a unified structure of this emerging field. Consequently, the general understanding on the principles of Byzantine machine learning remains poor. This article addresses this issue by presenting a primer on Byzantine machine learning. In particular, we introduce three pillars of Byzantine machine learning, namely the concepts of *breakdown point*, *robustness*, and *gradient complexity*, to curate the efficacy of a solution. The introduced systematization enables us to (i) bring forth the merits and limitations of the state-of-the-art solutions, and (ii) pave a clear path for future advancements in this field.

CCS Concepts: • **Computing methodologies → Machine learning**; **Distributed artificial intelligence**; • **General and reference → Surveys and overviews;**

Additional Key Words and Phrases: Byzantine machine learning, distributed SGD, robust aggregation

ACM Reference format:
Rachid Guerraoui, Nirupam Gupta, and Rafael Pinot. 2024. Byzantine Machine Learning: A Primer. *ACM Comput. Surv.* 56, 7, Article 169 (April 2024), 39 pages.
https://doi.org/10.1145/3616537

1 INTRODUCTION

Machine learning is arguably the main driving force behind the rapid growth of modern *Artificial Intelligence (AI)* technology [48, 167]. Accordingly, the race for designing more accurate and versatile AI systems has led to an unprecedented rise in the complexity of machine learning tasks. In particular, both the models and the training datasets are growing in size by the day, especially when considering complex tasks such as natural language processing [34, 82, 202]. To keep up with the growing computational demand of these machine learning tasks, it is now common to heavily rely upon distributed methodologies [1, 20, 54, 107, 221, 225]. This field is commonly referred to as *distributed machine learning*. Essentially, the training procedure is fragmented into several (simpler) sub-tasks that are distributed on different machines (or *nodes*) [149]. The nodes perform their respective sub-tasks in parallel and coordinate their local actions with each other,

This work was partly supported by Swiss National Science Foundation (SNSF) grant 200021_200477.
Authors' address: R. Guerraoui, N. Gupta, and R. Pinot, EPFL IC IINFCOM DCL, INR 311 (Bâtiment INR), Station 14, CH-1015 Lausanne, Vaud, Switzerland; e-mails: rachid.guerraoui@epfl.ch, nirupam115@gmail.com, rafael.pinot@epfl.ch.

0360-0300/2024/04-ART169 $15.00
https://doi.org/10.1145/3616537

either by using a trusted authority (usually referred to as *server*) [22, 123] or by interacting directly with each other [32]. The former scheme is sometimes called *server-based* coordination, and it includes the celebrated *federated learning* mechanism. As for the latter, it is often called *peer-to-peer* coordination. Many prominent distributed machine learning algorithms consist of distributing among the nodes the expensive task of gradient computations in the renowned *Stochastic Gradient Descent (SGD)* method. Such a distributed implementation of SGD, collectively referred to as *Distributed Stochastic Gradient Descent (D-SGD)*, divides the workload for each node by the total size of the system (i.e., the total number of nodes), thereby rendering the training of large complex models less cumbersome [22, 126]. Moreover, distributed machine learning algorithms do not require the nodes to share their respective data and therefore naturally provide the nodes some level of sovereignty over their training dataset. These appealing benefits of distributed machine learning have carried this field to the forefront of machine learning (and AI) research, in industry and academia alike [13, 17, 44, 63, 77, 87, 103, 127, 187, 197, 201].

1.1 Robustness Issues in Distributed Machine Learning

The ability of distributed machine learning to train large accurate models has substantially enlarged the range of possible applications of machine learning models. Besides traditional applications, such as handwriting recognition [157], speech to text conversion [190], and facial recognition [111], machine learning solutions can now be used in critical data-oriented public domain services like healthcare [169], banking/finance [60, 118], environmental science [222], cybersecurity [5, 99], and even the military [101, 141].

Nevertheless, a distributed machine learning paradigm is quite susceptible to "misbehaving"nodes [75, 180]. In particular, a deviation of a handful of nodes from the prescribed instructions of standard distributed machine learning algorithms can highly disrupt the learning procedure (e.g., see the experimental results in the work of Allen-Zhu et al. [10]). Such behavior can be caused by software and hardware bugs, by poisoned data (e.g., see [25, 134]), or by malicious players controlling part of the system (e.g., see [70]). Following the nomenclature of distributed computing, we find it convenient for the sake of generality to refer to such nodes as *Byzantine* [116]. The presence of such Byzantine nodes, which is arguably inevitable in a real-world distributed setting [16, 36, 131], impacts the trust on the learning efficacy of distributed machine learning solutions. But more importantly, this vulnerability may even lead to severe societal repercussions, mainly due to the recent proliferation of machine learning models, trained using distributed algorithms, in many sensitive public domain applications [19, 150, 163, 196]. Malicious entities could, for instance, target healthcare recommender systems being developed by means of distributed learning algorithms to manipulate public opinions on critical issues concerning public health [189]. Similar threats may arise when using distributed learning solutions for gathering military intelligence [2].

The growing usage of machine learning solutions in our daily life makes it more crucial than ever before to ensure the robustness of the underlying distributed algorithms against Byzantine actors [2, 64]. Not surprisingly, a rapidly growing topic of research, often coined *Byzantine machine learning*, has emerged. The overall objective is to design schemes that impart robustness to D-SGD methodologies against arbitrary actions of Byzantine nodes. Such schemes are quite appealing, as they can tolerate software and hardware bugs, poisoned data, and even malicious players controlling part of the system. In short, such a scheme makes *reliable machine learning* possible using *unreliable components*.

As of today, a significant number of Byzantine machine learning schemes have been proposed. More attention has been given to server-based coordination mechanisms [6, 10, 26, 31, 45, 67, 73, 92, 109, 205, 214] compared to their peer-to-peer counterparts [65, 66, 91, 100, 211]. The apparent disparity between the two classes of mechanisms can be partially explained by the simplicity of the problem in the server-based setting. Basically, the presence of a trusted coordinator (i.e., the server) eliminates any *drift* between the nodes' local models, which greatly limits the detrimental potency of a Byzantine actor. Nevertheless, studying Byzantine robustness of D-SGD in the server-based framework proves to be a pivotal step toward addressing the problem in the more complex peer-to-peer setting wherein the nodes do not have access to a trusted server [65, 100]. Several works have also presented empirical evaluations of the proposed solutions by designing real-world Byzantine behaviors (or attacks) [10, 18, 37, 70, 74, 79, 88, 108, 124, 170, 206]. These Byzantine simulations provided interesting experimental testbeds to (i) expose the weaknesses of traditional D-SGD methods in either server-based or peer-to-peer settings, and (ii) evaluate the robustness claims of the different schemes.

1.2 Robust Aggregation: A Key to Byzantine Robustness

Prominent Byzantine robustness schemes in D-SGD typically build upon a key element called *robust aggregation* [30, 67, 128, 171, 210]. To get an intuition on the critical role played by robust aggregation, let us quickly recall some basics of D-SGD. The main reason for the improved computational efficiency of D-SGD over its centralized counterpart is the use of multiple *local small oracles* that can be easily actualized by the nodes, as opposed to a *global big oracle*, which is generally quite costly to actualize in a large-scale machine learning task [22]. Next, we briefly describe this distributed methodology and its robustness flaws.

1.2.1 *D-SGD and Its Susceptibility to Byzantine Behaviors.*

The training procedure is decomposed into three phases, namely the *local computations*, *communication*, and the *global model update(s)*. Although the specific details of these phases may vary depending upon the underlying architecture of the communication network (i.e., server based or peer-to-peer), the main concepts remain the same. For pedagogical reasons, we summarize these phases in the following for the server-based setting wherein there is only one model (maintained by the server) at each training step (or iteration). The detailed description of these phases for both server-based and peer-to-peer settings can be found in Section 3.2. The server begins each iteration by broadcasting the current model to all nodes in the system. Then, each round proceeds as follows:

- *Local computations*: Each node draws a data point from its local data generating distribution to compute a stochastic estimate of the gradient (a.k.a. stochastic gradient) of its local loss function at the current model.
- *Communication*: Each node sends back to the server its computed stochastic gradient.
- *Global model update*: Upon receiving the gradients from all the nodes, the server *aggregates* them, using the simple averaging operation, to update the current model.

Although there is no centralized server, the averaging operation remains a crucial part of the peer-to-peer setting. However, this averaging scheme is rendered ineffective in the presence of Byzantine nodes. Specifically, as the average value of the gradients can be changed arbitrarily by changing just one gradient, even a single Byzantine node may arbitrarily manipulate the global model update by sending malicious incorrect gradients to the server [26]. This could significantly compromise the accuracy of the model delivered by the learning algorithm [10, 18, 92, 206]. Henceforth, we classify the set of nodes as Byzantines and non-Byzantines, also called *honest*.

1.2.2 *Robust Aggregation to Fix the Brittleness of D-SGD.*

According to the preceding, a crucial step toward Byzantine robustness in D-SGD is to replace the simple averaging operation by a more sophisticated *robust aggregation rule* that would protect the correctness of the global model updates from Byzantine perturbations. As of today, several such robust aggregation rules have

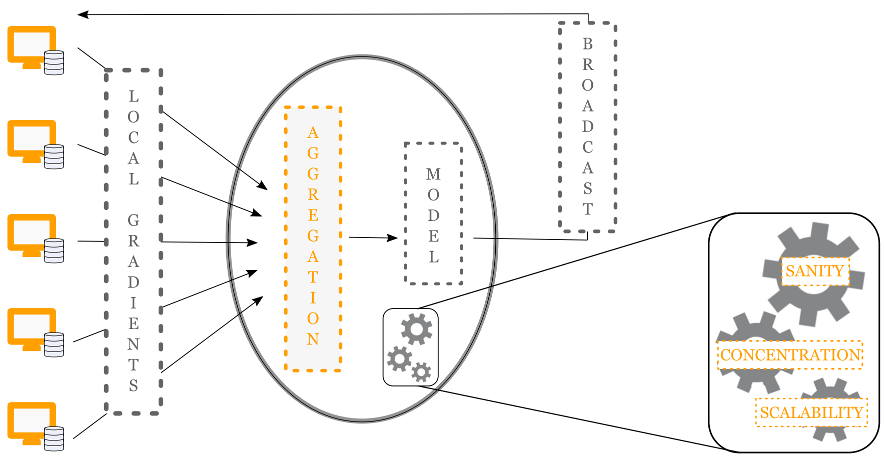

Fig. 1. Schematic of D-SGD in a server-based framework with a robust aggregation rule. Refer to Appendix A for details on the three key properties of sanity, concentration, and scalability.

been proposed. A common working principle behind these rules is to reduce the sensitivity of the aggregated output with respect to individual inputs, or a subset of them. For example, consider the *median* operator. Unlike averaging, the median of n values cannot be changed arbitrarily unless we change a majority of the values. By reducing sensitivity, a robust aggregation rule limits the ability of Byzantine nodes to disrupt the update procedure. However, at the same time, the aggregation rule should also yield meaningful model updates. Hence, we identify three key properties that a robust aggregation rule should satisfy, namely *sanity*, *concentration*, and *scalability*, detailed in Appendix A. The overall scheme of D-SGD with robust aggregation is illustrated in Figure 1.

A few alternatives to using the robust aggregation rule for Byzantine robustness include *suspicion*-based protocols [6, 10, 208] and *redundancy*-based protocols [43, 52, 94, 159]. These methods only apply to distributed settings in which the data held by the different nodes is either identical or *homogeneous*. They fail to provide any robustness (or even operate) when nodes hold different datasets (i.e., the data across the nodes is *heterogeneous*), which is indeed the case in practical distributed settings [65, 107, 108]. Moreover, error-coding-based approaches [43, 159] induce extreme latency in the system and enforce synchronicity in their execution. Others, namely the suspicion-based protocols [6, 10], rely upon strong assumptions on the data generating distributions that generally are violated in practice. Additionally, these approaches predominantly rely on a trusted server and need not apply to a peer-to-peer architecture.

However, robust aggregation rules have been shown to be applicable and effective in various different system settings, including heterogeneity [65, 92, 108], asynchrony [51, 65, 72], and different architectures [30, 90, 210]. Robust aggregation rules can also be easily used within communication-efficient [81, 207] and accelerated variants [109] of SGD. Such modularity of this approach makes it an indispensable asset in designing real-world distributed machine learning systems that are provably resistant to Byzantine failures.

1.3 This Survey: A Unifying Perspective on Byzantine Machine Learning

The goal of this survey is to make a pedagogical contribution to the important field of Byzantine machine learning. We focus on methods that attempt to confer Byzantine robustness to standard distributed machine learning algorithms through the use of robust aggregation. Upon closely examining the existing analyses of these methods, we have observed that their theoretical guarantees

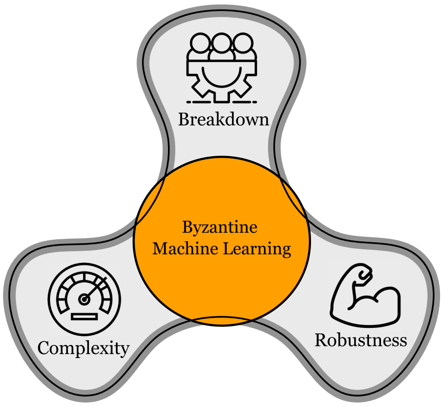

Fig. 2. Our novel unifying framework enables evaluation of Byzantine machine learning solutions based on the criteria of breakdown point, robustness, and gradient complexity.

rely upon a non-uniform set of assumptions and demonstrate disparate robustness properties for the aggregation rules. Such ruggedness in the field makes it rather difficult to compare the merits (or limitations) of the different Byzantine robustness schemes and obstructs any meaningful judgment on the shortcomings of the current solutions. To standardize the study of Byzantine machine learning solutions, we introduce a new framework for evaluating the performance of the proposed schemes. Our unifying framework for Byzantine machine learning, illustrated in Figure 2, builds upon three key concepts: *breakdown point*, *robustness*, and *gradient complexity*.

The notion of breakdown point has been used predominantly in the field of robust statistics to measure the ability of an estimator to resist outliers [104, 119]. Interestingly, this notion can be immediately adopted to Byzantine machine learning. We define the breakdown point of a distributed learning algorithm to be the maximum number of Byzantine nodes it can tolerate to ensure a bounded loss in the accuracy of the model. In other words, an algorithm Π may output a model with arbitrarily bad accuracy if the number of Byzantine nodes in the system are more than the breakdown point of Π. The second pillar, robustness, is the accuracy guaranteed by the learning algorithm in the presence of Byzantine nodes, provided the number of Byzantine nodes is less than the breakdown point of the algorithm. The third pillar, gradient complexity, addresses the computational overhead incurred by a Byzantine-resilient D-SGD scheme over the standard D-SGD. This overhead mostly consists of additional gradients that honest nodes must compute to overcome any potential degradation in the accuracy of the model due to malicious deviation of Byzantine nodes from the prescribed learning protocol [73, 109, 214]. The unavoidable impact on the gradient complexity of distributed machine learning due to Byzantine perturbations is analogous to that in robust statistics [104].

Note that the breakdown point for a learning algorithm can be computed in an offline manner—that is, it can be determined independent of the learning algorithm by solely analyzing the aggregation rule used in D-SGD. To analyze the other two properties (i.e., robustness and gradient complexity), we must examine the convergence of the resulting learning algorithm. Specifically, we decompose the error of the iterative algorithm, upon the completion of a pre-determined number of steps, into two parts: *vanishing error* and *non-vanishing error*. The vanishing error, as the name suggests, approaches zero as the number of iterations approach infinity, usually with a sublinear rate similar to SGD [80]. The non-vanishing error is the residual error that is independent of the number of iterations (i.e., it cannot be reduced by increasing the number of iterations) and is the smallest possible error the learning algorithm can obtain. Hence, the robustness property of the learning algorithm is quantified by the non-vanishing error. When the data held by the different nodes is assumed to be *homogeneous*, the non-vanishing term disappears for effective robust aggregation rules (i.e., we can obtain perfect robustness). However, this is not the case when the data is *heterogeneous*, which is often the case in practical distributed settings [107]. The vanishing error quantifies the number of iterations needed to guarantee a given convergence error. As the number of iterations is directly proportional to the number of (stochastic) gradients computed by each honest node, we can use the vanishing error to measure the gradient complexity of a solution. The introduction of these three criteria allows us to evaluate various Byzantine machine learning solutions from the literature, as well as identify the remaining challenges and gaps in the field.

1.4 Other Related Surveys

There are other surveys dedicated to efficient and secure applicability of machine learning solutions in specific public-domain tasks (e.g., healthcare [158], cybersecurity [5], and environmental science [222]); however, we primarily focus on the issue of Byzantine robustness in distributed training of generic machine learning models. Recent surveys [30, 170, 171] on Byzantine machine learning present a documentation of the many solutions that have been proposed as of now for imparting Byzantine robustness to D-SGD. The aim of our survey is to rather provide an in-depth understanding on the underlying principles of Byzantine machine learning and the upcoming trends in the field. Upon discussing recent results on fundamental limitations in this problem, we introduce a unifying system to provide insightful evaluation on the performance of existing solutions. Our introduced systematization can be applied for evaluating both the current and upcoming approaches to Byzantine machine learning. By improving the understanding of the literature, our survey also sheds light on the shortcomings of the state of the art and thereby motivates future topics in this important field. More details on the initial trends in Byzantine machine learning can be found in the work of Yang et al. [212].

1.5 Article Organization

We start by providing, in Section 2, an overview of other (potentially) related topics and discuss to what extent they can or cannot be linked to Byzantine machine learning. In Section 3, we present the necessary background on distributed machine learning, including the description of D-SGD in both server-based and peer-to-peer settings, as well as a discussion on the inaptness of this method to the Byzantine scenario. Then, in Section 4, we formally introduce the goal of Byzantine machine learning and present our three criteria of *breakdown point*, *robustness*, and *gradient complexity*. Section 5 discusses the correctness guarantees of existing solutions to Byzantine machine learning, and also presents a comparison on these schemes based on their breakdown point, robustness, and efficiency. Finally, we present concluding remarks and possible future directions in Section 6.

2 THE MANY FACES OF BYZANTINE MACHINE LEARNING

The problem of Byzantine robustness in distributed machine learning was first formally introduced by Feng et al. [75], who questioned the correctness of standard machine learning algorithms in distributed settings that may be infiltrated by misbehaving parties. Concurrently, the problem also received some attention from the closely related fields of distributed optimization [179–182, 184] and state estimation in distributed systems (e.g, [47, 137, 147, 174]). Meanwhile, the growing interest on robustness in machine learning has also been supported by the robust statistics community through design and analysis of new robust estimation schemes [23, 42, 56–58, 115, 156, 177]. In the following, we present comparisons between Byzantine machine learning and other related problems, and discuss the applicability of their solutions to Byzantine robustness in D-SGD.

2.1 Byzantine Robustness in Distributed Optimization

Some works [95, 180] have shown that generally it is impossible to solve a distributed optimization problem when some nodes may behave arbitrarily. Specifically, Gupta and Vaidya [95] proved the necessity of a redundancy property, named *$2f$-redundancy*, to obtain an optimal solution in the presence of f Byzantine nodes out of n total nodes. Essentially, the $2f$-redundancy property requires that the optimization problem (i.e., minimizing the average cost functions of honest nodes) should be equivalent to minimizing the average cost functions of only $n - 2f$ honest nodes. Under this redundancy property, they showed that we can modify the classical distributed gradient descent method (i.e., the deterministic counterpart of D-SGD) by replacing the averaging operation by a robust aggregation, called *Comparative Gradient Elimination (CGE)* [93, 95]. However, other works [180, 185] proved a similar result considering univariate cost functions using the aggregation rule of called *trimmed mean*. Moreover, Su and Vaidya [180] also presented the notion of *approximate* distributed optimization under Byzantine threats to analyze the problem in the scenario when the $2f$-redundancy property need not hold. Similar approximations were also studied for multivariate cost functions in the work of Liu et al. [129] and Su and Vaidya [179], where the authors analyzed the modification of distributed gradient descent using the *Coordinate-wise Trimmed Mean (CwTM)* and CGE aggregation rules. Additional Byzantine robustness by CwTM and CGE was studied for the peer-to-peer distributed optimization problem in subsequent years [91, 114]. These works in optimization, however, only consider convex or strongly convex cost functions, which is usually not the case in the context of machine learning. A quick review on additional works on Byzantine robustness in distributed optimization can be found in the work of Fanitabasi [71].

The issue of Byzantine robustness has also been studied for a special case of distributed optimization, namely for state estimation of systems [47, 135, 137, 139, 146, 147, 162, 174, 175, 178]. In this context, the objective is to reconstruct the true state of the system given the observations made by n sensors, out of which f may be Byzantine. Interestingly, the condition of $2f$-redundancy was independently proven to be necessary (but was called *$2f$-observability*) for reconstructing the system states under Byzantine observations [47, 147]. Most of the solutions proposed for this problem assume access to the entire set of observations (included the corrupted ones), which would be equivalent to having access to the entire dataset in the context of distributed machine learning [135, 146, 175]. Besides the impracticality of transferring the entire training dataset to a single machine, a Byzantine node in the distributed machine learning setting need not commit to any particular set of data points. In fact, in distributed machine learning, Byzantine nodes can send fabricated gradients or model updates [18, 206] that do not correspond to any dataset at all. Finally, state estimation of linear systems is mostly solved via quadratic programming [135, 139, 146, 162]—that is, the loss functions are quadratic strongly convex and the parameters are known *a priori*, which is seldom true in the context of machine learning. These peculiarities make it difficult to apply most of these techniques to Byzantine machine learning.

2.2 Robust Estimation with Arbitrarily Corrupted Data

The problem of robust estimation with corrupted data [23, 25, 42, 56–58, 115, 156, 177] can be treated as a variant of Byzantine machine learning where the identity of Byzantine nodes may change from one training step to another, and the honest nodes are assumed to always sample data from an identical distribution. However, many robust estimation schemes rely upon the assumption that the data is accessible, which is not the case in distributed machine learning [25, 56, 57, 156]. Additionally, they assume that the estimation problem can be modeled as a quadratic (or convex) programming scheme, which also is seldom the case in machine learning. Most modern-day machine learning problems are indeed non-convex. Although one-shot robust estimators such as *Geometric Median (GM)* [98, 176], *Minimum Diameter Averaging (MDA)* [166], and *median of means* (MoM) [106, 144] can be use as aggregation rules for robustifying D-SGD [4, 45, 67, 151], proving the convergence of the resulting algorithm is not straightforward and often requires additional appendages [109]. Many challenges need to be addressed to be able to adapt these kind of methods to a machine learning problem. Among them, we highlight the following. First, whereas most results studying robust statistics assume the probability distribution of the data to be circumscribed in a given class (e.g, uni-modal or sub-exponential), the distribution generating the inputs in a machine learning model is usually assumed to be arbitrary [3, 28, 29]. Second, the distributions characterizing the inputs of the honest nodes (e.g., their update vectors) need not be identical in machine learning [107]. In the robust statistics literature, the honest inputs are generally assumed to have identical distributions. Finally, the distributions of the honest nodes' inputs evolve over the course of the learning procedure [73], which is not usually the case in robust statistics. In other words, the distribution of the inputs is usually static (or *a priori* fixed) in robust estimation problems.

2.3 Byzantine Consensus

It is appealing to consider the viewpoint of distributed computing literature to present a formal treatment of Byzantine machine learning. However, it is important to understand the key differences between the objectives of Byzantine machine learning and the objectives of the classical Byzantine consensus problem in the field of distributed computing. In Byzantine consensus, each node starts with an input value and the goal is to design a distributed algorithm satisfying three properties, namely *termination*, *agreement*, and *validity* [36, 131]. The properties of termination and agreement require that each honest node outputs a value (in finite time) and that these output values should be identical, respectively. The third property of validity generally requires that the output value should be one of the honest nodes' input. In contrast to Byzantine consensus, in a Byzantine machine learning solution the honest nodes need not have identical output values (i.e., learning models) nor should their final models be equal to any of their initial models. In particular, in Byzantine machine learning, the honest nodes usually initialize their model with arbitrary values prior to the learning procedure and stop when they reach a critical point for their loss function; hence, applying such agreement and validity conditions would not be appropriate.

Nevertheless, we note that the primitives designed for solving the *iterative approximate Byzantine consensus* problem [117, 168, 191–195, 218, 219] have proved useful in designing some robust aggregation schemes for ensuring Byzantine robustness in distributed gradient descent [97, 178, 183, 211].

2.4 Connection with Evasion Attacks

Modern neural networks achieve state-of-the-art performance in a variety of domains. However, it has been shown that such neural networks can be vulnerable to evasion attacks [24, 186]. An evasion attack refers to a small, imperceptible change applied to an input data point that is maliciously designed to fool a machine learning algorithm. The vulnerability of state-of-the-art machine learning models to these attacks has major security implications, especially when models are used in AI-driven technologies, such as image recognition for autonomous cars or in the medical field. Accordingly, the problem of defending against such attacks also attracted considerable attention [15, 41, 49, 133, 136, 152–154, 188]. Although they might appear similar from a high-level viewpoint, it is worth noting that Byzantine robustness and robustness to evasion attacks are two orthogonal concepts. In particular, the robustness of a model (at testing time) to evasion attacks does not provide any guarantee about the robustness (at training time) to Byzantine behaviors. Similarly, as Byzantine robustness focuses on the training (optimization) procedure, we can always train models using a Byzantine-resilient aggregation rule but without obtaining robustness to evasion attacks. Characterizing the connection between these two notions of robustness remains an open problem.

3 BACKGROUND ON DISTRIBUTED MACHINE LEARNING

We consider a system with n nodes represented by indices $1, \dots, n$, and a space of data points $\mathcal{X}$. We assume that each node i has query access to an unknown data generating distributions $\mathcal{D}_i$ over $\mathcal{X}$. Each node i wishes to learn a machine learning model, parameterized by a real-valued vector $\theta^{(i)} \in \mathbb{R}^d$, that ensures optimal accuracy over the joint data distribution across all the nodes in the system. Specifically, for a given parameter vector θ and a data point $x \in \mathcal{X}$, the model incurs a real-valued loss $q(\theta, x)$. Accordingly, each node i defines a local loss function to be

$$Q_i(\theta) = \mathbb{E}_{x \sim \mathcal{D}_i}\left[q(\theta, x)\right] \quad \forall \theta \in \mathbb{R}^d. \tag{1}$$

We assume q to be differentiable with respect to θ (with continuous derivative) and each loss function Q_i to be L-smooth [29]. Each node aims to compute a parameter vector $\theta_*^{(i)} \in \mathbb{R}^d$ that minimizes the collaborative loss function. In other words,

$$\theta_*^{(i)} \in \underset{\theta \in \mathbb{R}^d}{\operatorname{argmin}}\, Q(\theta); \quad Q(\theta) = \frac{1}{n}\sum_{i=1}^{n} Q_i(\theta) \quad \forall \theta \in \mathbb{R}^d. \tag{2}$$

We assume that the function Q admits a minimum—that is, $\min_{\theta \in \mathbb{R}^d} Q(\theta)$ exists and has a finite value. However, as Q could be non-convex (e.g., when considering deep neural networks), solving the preceding optimization problem is NP-hard in general [33]. Therefore, a more reasonable and widely accepted goal for each node is to compute a critical (or stationary) point of Q—that is, $\theta_*^{(i)}$ such that $\|\nabla Q(\theta_*^{(i)})\| = 0$, where ∇Q is the gradient of Q and $\|\cdot\|$ denotes the Euclidean norm. When Q is a convex function, any critical point is indeed a minimum.

To solve the preceding optimization problem, each node must collaborate with other nodes in the system, due to the differences in the data distributions [32]. In the case when the data distributions are identical, collaboration is still beneficial since it significantly reduces the individual computational costs for each node and renders the learning procedure less cumbersome [22, 29, 126]. In the following, we mention the different architectures nodes can utilize for collaboration, followed by descriptions of the different implementations of D-SGD.

3.1 System Architectures

Recall that each round of a D-SGD algorithm proceeds in three phases [22, 113]: *local computations*, *communication*, and *global model update*. The communication mode plays a key role in inter-node collaboration, which is crucial to building an accurate global model. The communication phase during the learning procedure is determined, to a great extent, by the system architecture. The different system architectures in standard distributed machine learning algorithms can be classified into two main categories [20, 197]: *server based* and *peer-to-peer*. We illustrate the two architectures in Figure 3 and further describe them next:

(1) In a server-based architecture, the nodes communicate through a (trusted) central server. The server plays the role of coordinating the local actions taken by the individual nodes. Note that a server can itself be viewed as an abstraction of a cluster of machines that enables coordination [123].

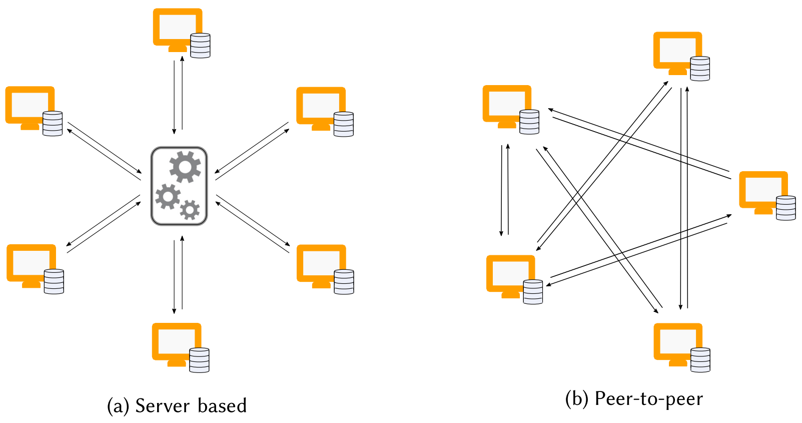

Fig. 3. Server-based and peer-to-peer system architectures with six and five nodes, respectively.

(2) In a peer-to-peer architecture, the nodes interact directly with each other without any central server. The underlying communication network is modeled by a graph representing the possible communication paths. Essentially, a node sends messages to another node only if there is an edge connecting the two in the direction of the communication.

We illustrate in Section 3.2 how the architecture influences the implementation of a learning scheme by presenting the implementations of D-SGD in both of the aforementioned architectures.

Remark 1. For pedagogical reasons, we only consider synchronous executions. This considerably simplifies the presentation of their operations, and we can precisely introduce elements that are essential for understanding the threat of Byzantine actors. Indeed, most existing works in Byzantine machine learning consider this setting, with a few exceptions [51, 65, 72]. Importantly, the key efficacy metrics (that we present shortly) for Byzantine machine learning generally remain the same even under the loss of synchronicity.

3.2 Distributed Stochastic Gradient Descent

The preceding distributed learning problem can be easily solved (in the absence of Byzantine workers) by using the D-SGD scheme. This is an iterative scheme that constitute the building block of many distributed machine learning systems [20]. The general principles of D-SGD do not depend on the system architecture; however, the details of the algorithm may change. In the following, we describe D-SGD in detail for both server-based and peer-to-peer architectures.

3.2.1 D-SGD in a Server-Based Architecture. The principal advantage of the server-based architecture is that, by coordinating the communication, the server can eliminate the potential drift between the local models. The server starts by initializing a parameter vector θ_0. This can be done at random or by using expert knowledge on the target model [142]. In each iteration t, the server maintains a parameter vector θ_t that is broadcast to all the nodes. Each node $i \in \{1, \ldots, n\}$ updates its current local parameter vector as $\theta_t^{(i)} = \theta_t$. The iteration proceeds in three phases described as follows [22] and illustrated in Figure 4:

- *Local computations*: Given a current model $\theta_t^{(i)}$, each node i computes a stochastic estimate $g_t^{(i)}$ of the gradient $\nabla Q_i(\theta_t^{(i)})$. Specifically, $g_t^{(i)}$ is computed by drawing a data point x

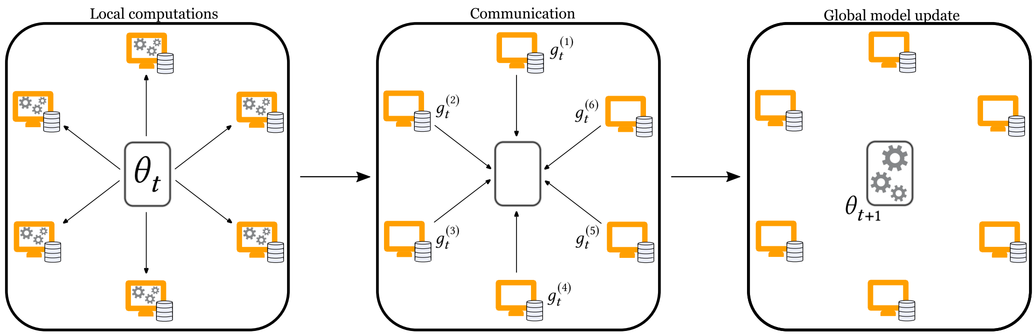

Fig. 4. Schematic of an iteration of D-SGD in the server-based setting.

(or sample) from the local data generating distribution $\mathcal{D}_i$ as follows[1]:

$$g_t^{(i)} = \nabla q\left(\theta_t^{(i)}, x\right), \text{ with } x \sim \mathcal{D}_i. \tag{3}$$

Note that, owing to (1), $g_t^{(i)}$ equals the true gradient $\nabla Q_i(\theta_t^{(i)})$ in expectation.

- *Communication*: Each node i sends to the server a message containing its computed stochastic gradient $g_t^{(i)}$.
- *Global model update*: Upon receiving the messages from all the nodes, the server computes the average of all the received gradients (i.e., $\hat{g}_t = \frac{1}{n}\sum_{i=1}^{n} g_t^{(i)}$). Then, the server updates the current model parameter vector θ_t to θ_{t+1} using a *learning rate* $\gamma_t > 0$. Specifically,

$$\theta_{t+1} = \theta_t - \gamma_t \, \hat{g}_t. \tag{4}$$

The preceding iterative procedure is repeated until a stopping criteria is met. Typically, this can be parameterized by a maximum number of iterations T—that is, the algorithm terminates after T iterations.

3.2.2 D-SGD in a Peer-to-Peer Architecture. We model the underlying communication network by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ whose set of vertices $\mathcal{V} = \{1, \ldots, n\}$ represent the nodes and set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represent the direct communication links between the nodes. In other words, node i can send direct messages to node j if and only if $(i, j) \in \mathcal{E}$. For each node i, we denote by Γ_i^{out} and Γ_i^{in} the sets of outgoing and incoming neighbors, respectively (i.e., $\Gamma_i^{out} = \{j \in \mathcal{V} \; ; \; (i, j) \in \mathcal{E}\}$ and $\Gamma_i^{in} = \{j \in \mathcal{V} \; ; \; (j, i) \in \mathcal{E}\}$). As convention, we commonly assume that $(i, i) \in \mathcal{E}$ for all $i \in \mathcal{V}$. In this particular framework, each node i initializes the learning procedure independently by choosing a local model $\theta_0^{(i)}$ and updates it iteratively by interacting with its neighboring nodes. In each iteration t, each node i maintains a local model $\theta_t^{(i)}$ that is updated using the following three phases [126], illustrated in Figure 5:

- *Local computations*: Each node i computes a stochastic gradient $g_t^{(i)}$ of its local loss function's gradient at $\theta_t^{(i)}$ (i.e., $\nabla Q_i(\theta_t^{(i)})$) identically to (3) in the server-based setting, and updates its current local parameter to $\theta_{t+1/2}^{(i)}$ using a local learning rate of $\gamma_t^{(i)}$. Specifically,

$$\theta_{t+1/2}^{(i)} = \theta_t^{(i)} - \gamma_t^{(i)} \, g_t^{(i)}. \tag{5}$$

[1]The nodes may also use a mini-batch of data points.

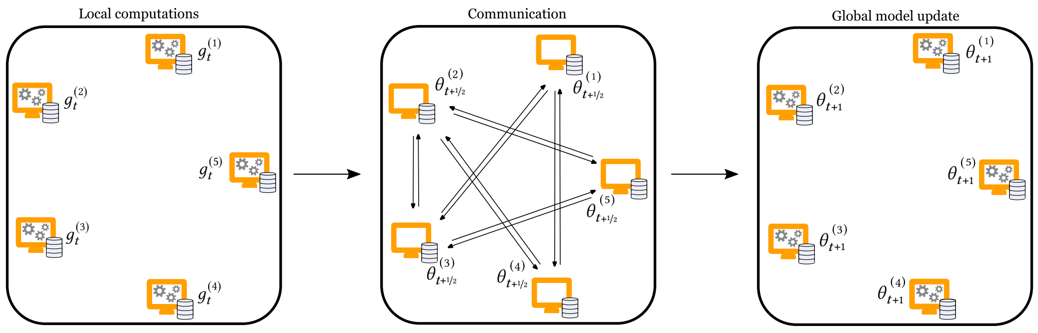

Fig. 5. Schematic of an iteration of D-SGD in the peer-to-peer setting.

- *Communication*: Each node i broadcasts a message containing $\theta_{t+1/2}^{(i)}$ to all of its outgoing neighbors (i.e., to all j such that $(i, j) \in \mathcal{E}$).
- *Global model update*: Upon receiving messages from all of its neighbors, each node i updates their partially update local parameter vector $\theta_{t+1/2}^{(i)}$ to $\theta_{t+1}^{(i)}$ by averaging all the received parameter vectors[2]—that is,

$$\theta_{t+1}^{(i)} = \frac{1}{|\Gamma_i^{in}|} \sum_{j \in \Gamma_i^{in}} \theta_{t+1/2}^{(j)}, \tag{6}$$

  where $|\cdot|$ denotes the set cardinality.

Similar to the server-based architecture, the preceding iterative procedure is repeated until a stopping criteria (e.g., after a pre-determined number of total iterations T).

3.2.3 Convergence of D-SGD in the Vanilla Setting. When all the nodes follow the prescribed instructions correctly (a.k.a. the vanilla setting), the preceding two distributed implementations of D-SGD provably converge to a critical point of the loss function Q (which need not to be convex), under standard assumptions on the boundedness of data heterogeneity [113, 126] and the local uncertainty in the stochastic gradients [80]. In the peer-to-peer case, some additional assumptions need to be made on the expansion of the communication graph $\mathcal{G}$ (see [113, 126, 143]). In particular, after running the D-SGD algorithm in either of the architectures for total T iterations, assuming appropriate learning rates and a complete network topology in the peer-to-peer case, we can obtain that [22, 126], for all nodes i,

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\left\|\nabla Q\left(\theta_t^{(i)}\right)\right\|^2\right] \leq \varepsilon, \quad \text{where } \varepsilon \in O\left(\frac{1}{\sqrt{nT}}\right). \tag{7}$$

The primary advantage of D-SGD over its centralized counterpart is the increased computational efficiency. As shown previously, to obtain the same convergence guarantees as its centralized counterpart, D-SGD divides the computational workload (i.e., the total gradients computations) of each node by the total size n of the system. Similar analysis can also be derived for the peer-to-peer setting with incomplete communication network topology [126, 143, 198]. In that case, however, the degree of efficiency also depends on the connectivity (or Laplacian spectrum) of the graph $\mathcal{G}$ and not just the total number of nodes [126].

[2]The nodes may also use weighted averaging for faster convergence [113].

The D-SGD scheme described previously has the potential of training large complex models; however, it also renders the training procedure vulnerable to misbehaving nodes, which are commonly modeled as Byzantine in distributed systems [116]. Specifically, we consider a scenario where the system may comprise up to f Byzantine nodes of *a priori* unknown identities. The presence of such nodes is highly common in large-scale distributed networks, owing to software/hardware bugs, data corruptions, or malicious players corrupting a part of the system [16]. In general, by modeling a distributed system with Byzantine nodes, we can analyze the worst-case scenarios for a distributed algorithm and eventually provide strong robustness guarantees. In the context of D-SGD, Byzantine nodes can arbitrarily manipulate the learning procedure by sending arbitrary misleading messages in the communication phase of the iterative scheme [26, 75, 170].

4 BYZANTINE MACHINE LEARNING

In this section, we review the goal of Byzantine machine learning, as well as the three criteria of *breakdown point*, *robustness*, and *gradient complexity*. These criteria enable us to curate the robustness of different existing augmentations of D-SGD, discussed later in Section 5, to Byzantine nodes.

4.1 Learning Objective in the Presence of Byzantine Nodes

In the presence of Byzantine nodes, the original objective of seeking a critical point (or a minimum) of the average loss function defined over all the nodes in the system cannot be met anymore [180]. A more reasonable goal is to find a critical point of the average loss function defined over only the honest nodes [95]. Accordingly, the goal of Byzantine resilient distributed machine learning, or *Byzantine machine learning*, is to design an algorithm that allows all the honest nodes to compute an accurate model over their collective data distribution, despite the presence of Byzantine nodes in the system. Specifically, let $\mathcal{H} \subseteq \{1, \dots, n\}$ represent the set of honest nodes where $|\mathcal{H}| \geq n - f$, and denote by $Q_{\mathcal{H}}(\theta)$ the average of honest nodes' loss functions—that is,

$$Q_{\mathcal{H}}(\theta) = \frac{1}{|\mathcal{H}|} \sum_{i \in \mathcal{H}} Q_i(\theta), \quad \forall \theta \in \mathbb{R}^d.$$

Then, the goal of Byzantine machine learning is formalized by the property of (f, ε)-Byzantine resilience [11], where recall that f denotes the total number of nodes that may be Byzantine.

Definition 1. For $\varepsilon \in \mathbb{R}_{\geq 0}$, a distributed algorithm is said to ensure (f, ε)*-Byzantine resilience* if, despite the presence of up to f Byzantine nodes, it enables each honest node $i \in \mathcal{H}$ to compute $\widehat{\theta}_i \in \mathbb{R}^d$ such that $\|\nabla Q_{\mathcal{H}}(\widehat{\theta}_i)\|^2 \leq \varepsilon$.

In practice, where nodes may only compute noisy estimates of the actual gradients of their local loss functions, similar to the non-Byzantine case, we acquiesce to stochastic stationarity—that is, requiring each honest node $i \in \mathcal{H}$ to compute $\widehat{\theta}_i$ such that $\mathbb{E}\left[\|\nabla Q_{\mathcal{H}}(\widehat{\theta}_i)\|^2\right] \leq \varepsilon$, where the expectation is defined over the randomness of data sampling (operated for computing stochastic estimate of true gradients) by the nodes.

4.2 Breakdown Point, Robustness, and Gradient Complexity

To measure the performance of a given distributed learning algorithm Π, in the presence of Byzantine nodes, we use three metrics: *breakdown point*, *robustness*, and *gradient complexity*, formalized next.

4.2.1 Breakdown Point. The breakdown point measures the tolerance limit of a distributed learning algorithm to Byzantine nodes, formally defined as follows (similar to robust statistics [104]). Basically, the breakdown point of a distributed learning algorithm is the (open) upper bound on f/n (i.e., the maximum possible fraction of Byzantine nodes) for which the algorithm can ensure (f, ε)*-Byzantine resilience* for some ε.

Definition 2 (Breakdown Point). The breakdown point of a given distributed learning algorithm Π is the minimum fraction BP_Π such that if $f/n \geq \mathsf{BP}_\Pi$, then Π cannot ensure (f, ε)-Byzantine resilience for any finite value of ε.

4.2.2 Robustness and Gradient Complexity. For introducing the other two metrics (i.e., robustness and gradient complexity), we focus our attention to a first-order iterative distributed learning algorithm Π_T, involving T iterations. Suppose that f/n is less than the breakdown point of Π_T, and thus Π_T ensures (f, ε)-Byzantine resilience for some ε. The error ε is an additive composition of two parts, the *vanishing error* and the *non-vanishing error*, which we denote by ε_T and ε_∞, respectively [108]. Specifically, $\varepsilon := \varepsilon_T + \varepsilon_\infty$. As the names suggest, the term ε_T is a decreasing function to T such that $\varepsilon_T \to 0$ as $T \to \infty$, whereas ε_∞ is invariant of T and remains unchanged even when $T \to \infty$. The robustness of Π_T is measured by ε_∞ (i.e., the best possible error Π_T can achieve), forcing an infinite amount of gradient computations from the honest nodes. Formally, we define robustness in this context as follows.

Definition 3 (Robustness). A first-order distributed learning algorithm Π_T is said to have ε_∞-*robustness* at f if Π_T ensures (f, ε_∞)-Byzantine resilience asymptotically (i.e., when $T \to \infty$).

Then, the gradient complexity is measured by ε_T—that is, the number of iterations sufficient to reach an error within a desired threshold from ε_∞. However, this measure may be misleading when the cost of gradient computations in a single iteration varies greatly from one algorithm to another. For instance, in the case of D-SGD (described in Section 3.2), each node computes a single stochastic gradient (by sampling only one data point) in each iteration. Yet, in the case of *mini-batch* D-SGD, the gradient estimates are computed by sampling multiple data points—that is, each node computes multiple stochastic gradients in each iteration. For consistency in complexity measure, and to incorporate solutions that include *dynamic sampling* for Byzantine resilience (e.g., see [65]), we substitute the number of iterations by the number of gradient computations per honest node as a measure of complexity.

Definition 4 (Gradient Complexity). Consider a first-order distributed learning algorithm Π_T with breakdown point BP_{Π_T} and robustness ε_∞. For a tuple $(f, \varepsilon_T) \in \mathbb{N} \times \mathbb{R}_{>0}$, with $f/n < \mathsf{BP}_{\Pi_T}$, the *gradient complexity* of Π_T is the minimum number of gradient computations sufficient per honest node to ensure $(f, \varepsilon_T + \varepsilon_\infty)$-Byzantine resilience.[3]

4.3 Known Limitations of Byzantine Machine Learning

We summarize in the following some known limitations on breakdown point, robustness, and gradient complexity applicable to a first-order Byzantine machine learning solution [96, 108, 128, 129]. These results are instrumental in design and analysis of robust aggregation rules for conferring Byzantine robustness to D-SGD [65, 72, 108, 129].

4.3.1 Breakdown Point. As formally shown by Liu et al. [129], no distributed learning algorithm (including higher-order optimization methods) can achieve a breakdown point greater than $1/2$.

[3]Note that the gradient complexity of Π_T is vacuous for a tuple (f, ε_T) where either $f/n \geq \mathsf{BP}_{\Pi_T}$ or $\varepsilon_T \leq 0$.

In other words, if half or more nodes are Byzantine, then we cannot obtain any guarantee on the approximation of the critical point. However, under special circumstances when the server has access to a verified part of the collective training dataset, as shown by Cao and Lai [39] and Regatti et al. [161], we can tolerate more than half Byzantine nodes. In the peer-to-peer case (i.e., when no trusted server is available to enable coordination among the nodes), using the standard impossibility result in distributed consensus [131], it is shown by Su and Vaidya [180, 183] that the breakdown point is smaller than or equal to $1/3$.

4.3.2 *Robustness.* It is shown by Gupta and Vaidya [95, 96] that no distributed algorithm can obtain *exact* robustness (i.e., $\varepsilon_\infty = 0$) unless the loss functions (i.e., the data distribution) of the nodes are correlated—specifically, they have $2f$*-redundancy*. This condition holds trivially when assuming homogeneity of data—that is, $\mathcal{D}_i = \mathcal{D}_j$ for any pair of honest nodes $i, j \in \mathcal{H}$. Other notions of data redundancy have also been shown sufficient (but not necessary) [42, 128, 179]. In the absence of such redundancy, we can characterize robustness under different models of data heterogeneity that bound the differences between the gradients of honest nodes' individual loss functions. The most widely studied heterogeneity model in the context of Byzantine machine learning is *G-gradient dissimilarity*, first studied in the work of Karimireddy et al. [108]. Under G-gradient dissimilarity, we assume that there exists $G \in \mathbb{R}_{\geq 0}$ such that for all $\theta \in \mathbb{R}^d$, $\frac{1}{|\mathcal{H}|} \sum_{i \in \mathcal{H}} \|\nabla Q_i(\theta) - \nabla Q_{\mathcal{H}}(\theta)\|^2 \leq G^2$. Under this particular heterogeneity model, as shown by Karimireddy et al. [108], we cannot achieve an arbitrarily small stationarity error, specifically $\varepsilon_\infty \in \Omega(\frac{f}{n} G^2)$. This bound has been shown to be tight in the work of Allouah et al. [11]. An alternate, and perhaps more reasonable, heterogeneity model is (G, B)*-gradient dissimilarity* [110], where we do not impose a uniform bound on the diversity of local gradients but rather let the difference grow with the norm of the average gradient of honest nodes. Specifically, we assume that there exists $G, B \in \mathbb{R}_{\geq 0}$ such that for all $\theta \in \mathbb{R}^d$, $\frac{1}{|\mathcal{H}|} \sum_{i \in \mathcal{H}} \|\nabla Q_i(\theta) - \nabla Q_{\mathcal{H}}(\theta)\|^2 \leq G^2 + B^2 \|\nabla Q_{\mathcal{H}}(\theta)\|^2$. This particular heterogeneity model has been extensively studied in the non-Byzantine setting (e.g., see [110, 113, 138, 145]). However, the work on Byzantine machine learning under (G, B)-gradient dissimilarity remains sparse with the only known upper bound on robustness provided in the work of Karimireddy et al. [108].

4.3.3 *Gradient Complexity.* There exist lower bounds on statistical rates (i.e., the number of data points sampled by each honest node) for a desired error ε in the homogeneous case (when $\varepsilon_\infty = 0$) (e.g., see [214, 223]). However, no formal work exists yet on characterizing the lower bound on gradient complexity in the context of first-order Byzantine machine learning. The task is especially challenging when considering the general class of non-convex loss functions.

5 AUGMENTING D-SGD FOR BYZANTINE MACHINE LEARNING

Recall (from Section 3.2) that D-SGD, in either server-based or peer-to-peer architecture, relies upon the averaging of local computations that nodes share over the network in the communication phase. Such an averaging operation is extremely fragile against incorrect (or corrupted) local computations that may be shared by Byzantine nodes. Indeed, vanilla D-SGD cannot sustain even a single Byzantine node [26]. In this section, we discuss the prominent adaptations of D-SGD proposed for solving the problem of Byzantine machine learning. We classify these solutions into three categories: (i) D-SGD with robust aggregation, (ii) D-SGD with variance reduction and robust aggregation, and (iii) D-SGD with momentum and a two-step aggregation scheme. In Table 1, we summarize the performance that can be achieved by these three categories of methods in terms of breakdown point, robustness, and gradient complexity.

Table 1. Summary of the Best Results Obtained for Byzantine Resilience of D-SGD Augmented Using the General Schemes Described in Section 5

General scheme	Breakdown point	Robustness (ID/H)	Gradient complexity	References
D-SGD with robust aggregation[(*)]	$\frac{1}{2}$	Optimal (ID)	$\times$	[26, 67, 205]
+ Variance reduction	$\frac{1}{2}$	Optimal (ID)	$O((\frac{f+1}{n} + \zeta)\,\frac{1}{\varepsilon^2})$[(**)]	[11, 73, 85, 109]
+ Two-step aggregation	$\frac{(1+\eta)}{2}$ with $\eta > 0$	Optimal (ID/H)	$O(\frac{f+1}{n}\,\frac{1}{\varepsilon^2})$	[11, 108]

[(*)] These results are obtained by making additional (non-trivial) assumptions on the stochastic nature of the gradients and sometimes on the fraction of Byzantine nodes in the system. Furthermore, these works present asymptotic convergence results, which hinders the analysis of the algorithms' gradient complexity.
[(**)] The meaning of ζ is discussed in Section 5.2. Also see the work of Allouah et al. [11] and Farhadkhani et al. [73] for more details.
Each class of method is evaluated in terms of breakdown point, robustness, and gradient complexity. "ID" means that the corresponding work assumes identical data distributions for all nodes. "ID/H" means that the work considers data heterogeneity but the results also apply to the homogeneous setting.

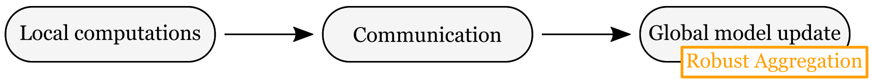

Fig. 6. D-SGD with robust aggregation only changes the model update phase. The first two phases of local computations and communication remain unchanged for the honest nodes.

5.1 D-SGD with Robust Aggregation

The main point of vulnerability of D-SGD lies in the averaging scheme used during the global model update phase. Indeed, this step can be arbitrarily manipulated by a single Byzantine node, which implies that D-SGD cannot be (f, ε)-Byzantine resilient for any $f > 0$ and $\varepsilon < \infty$ [26]. Therefore, early work on solving the Byzantine machine learning problem focused primarily on robustifying this phase by replacing the averaging by a robust aggregation rule. This adaptation is illustrated in Figure 6 and summarized formally in the following.

5.1.1 Integrating a Robust Aggregation in D-SGD. Next, we summarize the modifications in the two implementations of D-SGD (i.e., server based and peer-to-peer), where F denotes an aggregation rule that takes local computations shared by the nodes as inputs and outputs a vector in $\mathbb{R}^d$.

- *Server-based architecture*: In the last phase (i.e., the global model update phase from Section 3.2.1), instead of the average $\hat{g}_t$ of the local gradients sent by the nodes, the server uses $R_t := F(g_t^{(1)}, \ldots, g_t^{(n)})$ to update the current model as $\theta_{t+1} = \theta_t - \gamma_t R_t$. Although the gradient $g_t^{(i)}$ sent by an honest node i follows (3), a Byzantine node j might send an arbitrary vector for $g_t^{(j)}$.
- *Peer-to-peer architecture*: In the global model update phase (see Section 3.2.2), each honest node i computes $R_t^{(i)} = F(\theta_{t+1/2}^{(i)}, \theta_{t+1/2}^{(j)}, j \in \Gamma_i^{in} \setminus \{i\})$ and updates its local parameter vector to $\theta_{t+1}^{(i)} = R_t^{(i)}$. The local parameter vector $\theta_{t+1/2}^{(j)}$ for a Byzantine node j might be arbitrary and need not follow (5).

5.1.2 Overview of Existing Robust Aggregation Rules. Since the first attempt toward Byzantine-resilient machine learning [75], several aggregation rules have been proposed to replace the averaging scheme of D-SGD to provide protection against Byzantine nodes. Essentially, the role of a robust aggregation rule is to reduce the *sensitivity* of the method so that a small subset of inputs cannot arbitrarily manipulate the output [67, 73]. That way, the Byzantine nodes, usually assumed to be in the minority, have limited adversarial power over the learning procedure. Besides reduced sensitivity to outliers, the output of a robust aggregation rule should be *concentrated* around the true average for the algorithm to deliver a reasonably accurate model.

Many aggregation rules satisfying the preceding properties have been inspired by solutions to the well-known *robust estimation* problem in the field of robust statistics [104]. Among the most notable ones, we find the coordinate-wise aggregation rules, including CwTM [69, 164, 211, 214], *Coordinate-wise Trimmed Median (CwMed)* [31, 214], or *Mean of Medians (MeaMed)* [205]. An alternate approach to coordinate-wise techniques is the use of "geometric center," such as GM [4, 4, 45, 76, 151], MDA [67, 166], *Krum* [26], and *Multi-Krum* or *Bulyan* [67].[4] Additionally, some new robust aggregation rules have been designed and analyzed as norm filtering/clipping techniques. These methods include *Centered Clipping (CC)* [83] and CGE [81, 91–93, 220]. It is worth noting that, beyond reduced sensitivity and concentration, the aggregation should be scalable to not render the learning procedure infeasible. In particular, the worst-case computational complexity of the scheme should not become prohibitive as n or f grow. Unfortunately, some techniques based on "geometric centers" such as GM or MDA are known to be very computationally expensive and have poor scalability [166].[5] Details on the preceding aggregation rules, and the key properties of *concentration*, *scalability*, and *sanity*, can be found in Appendix A.

Remark 2. We do not attempt to provide an exhaustive list of methods (which can be found in the work of Bouhata and Moumen [30]). Instead, we present some notable schemes that have been analyzed by several works in recent years; they are therefore good representatives of the trends and advancements in the field over time.

5.1.3 Breakdown, Robustness, and Gradient Complexity. Existing analyses of D-SGD with robust aggregation mainly focused mainly on one criterion of Byzantine resilience, specifically the breakdown point. The robustness and gradient complexity were either largely ignored or suboptimal. We summarize the existing results on D-SGD with robust aggregation as follows:

- For schemes such as Krum [26], Multi-Krum [67], MDA [67], and MeaMed [205], we only have guarantees on the asymptotic convergence of the algorithm. The analyses assume the data distribution on all the nodes to be identical. The robustness and gradient complexity of these algorithms under heterogeneity remain unclear.
- For GM [45], CwTM [214], and CwMed [214], the convergence is analyzed assuming non-stochastic gradients where the data points for each node are assumed to be i.i.d. from a common ground truth distribution. Moreover, the learning problem is assumed convex. The gradient complexity of these algorithms in the non-convex stochastic case is unknown, and the robustness under heterogeneity remains unclear.
- The analysis of CGE relies on homogeneity (i.e., identical local data distributions) and strong convexity [92, 220]. Besides, CGE might only tolerate a small fraction of Byzantine nodes depending upon the *condition number* of the optimization problem characterizing the distributed learning task.

Note that the Byzantine resilience of CC, originally proposed by Karimireddy et al. [109], has been analyzed only when the nodes incorporate *gradient momentum* in their local computations,

[4]Bulyan is a meta-algorithm that aims to improve the performance of any given aggregation rule.
[5]GM does not have a closed-form solution [50, 176] and is often replaced by approximate practical variants such as *robust federated averaging* [151].

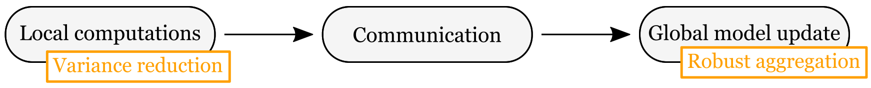

Fig. 7. D-SGD with robust aggregation at the update phase and a variance reduction scheme at the local computations phase.

which we discuss next in Section 5.2. Accordingly, we defer the discussion on breakdown, complexity, and robustness of CC to the next section. Although the analysis of CC by Gorbunov et al. [83] does not incorporate momentum, the aggregation rule is coupled with a *detection mechanism* (to detect Byzantine behaviors), which requires the set of training data points at each honest node to be identical.

It is important to note that the convergence of these robust variants of D-SGD rely on non-classical assumptions, such as (strong) convexity [45, 92, 211, 214], vanishing variance of the stochastic gradients [26, 51, 67], and sub-exponential/Gaussian data distribution [45, 76, 83, 214]. These assumptions are not common in the context of machine learning, which limits the applicability of these methods in practice [73, 109]. Last, some works [81, 91, 214] focus on *empirical loss* minimization—that is, the data distributions are assumed uniform over a finite number of data points, and they assume non-stochastic gradient computations.

5.2 D-SGD with Variance Reduction and Robust Aggregation

Proving Byzantine resilience of D-SGD with robust aggregation rules turns out to be challenging, resulting in the use of the non-classical assumptions mentioned earlier. This, shown in the work of Karimireddy et al. [109], is a fundamental limitation of *memoryless* aggregation due to their *permutation invariance property*. In other words, the trajectory of the parameter vector over the course of D-SGD with a memoryless aggregation remains invariant to arbitrary shuffling of the indices of the nodes across iterations. The use of *history* of past gradients breaks the permutation invariance property and is shown to be effective for Byzantine resilience in the work of Alistarh et al. [6] for convex learning problems. Similar observation is made for non-convex problems in the work of Allen-Zhu et al. [10]. However, these works assume *bounded support* of the data distributions, which is difficult to satisfy in machine learning [109]. More reasonable adaptations using *Polyak's momentum* [155] have been proposed by Farhadkhani et al. [73] and Karimireddy et al. [109]. The idea of momentum is also empirically shown to be effective [68, 165]. The critical property of momentum that is instrumental to overcoming the non-classical assumptions is the phenomenon of *variance reduction* [73]—that is, the noise in the local momentums of the nodes reduces over the course of the learning procedure and eventually vanishes (i.e., when $T \to \infty$), unlike the noise in the stochastic gradients. Another technique of variance reduction that is shown to be effective is *dynamic sampling* [65], but it comes at the expense of gradient complexity. Other variance reduction techniques, specifically SAGA [55] and VR-MARINA [84], have also been shown to be effective in the work of Gorbunov et al. [85] and Wu et al. [203]. We illustrate the general idea of these modifications in Figure 7 and detail in the following the methods of momentum and dynamic sampling.

5.2.1 *D-SGD with Distributed Momentum and Robust Aggregation.*

The momentum operation is incorporated in the local computation phase (see [73]). Specifically, in the local computations phase of each iteration t, each honest node i computes the *Polyak's momentum* of its stochastic gradient, denoted by $m_t^{(i)}$ and defined to be

$$m_t^{(i)} = \beta m_{t-1}^{(i)} + (1-\beta) g_t^{(i)}, \tag{8}$$

where $m_0^{(i)} = 0$ by convention, $\beta \in [0, 1)$ is the *momentum parameter*, and $g_t^{(i)}$ is the stochastic gradient as defined in (3). The consequent modifications to D-SGD in the different architectures are as follows.

- *Server-based architecture*: In the communication phase, each honest node i sends $m_t^{(i)}$, instead of $g_t^{(i)}$, to the server. In the global model update phase, the server computes $R_t := F(m_t^{(1)}, \dots, m_t^{(n)})$ to update the model as $\theta_{t+1} = \theta_t - \gamma_t R_t$. Recall that in the preceding, $m_t^{(i)}$ corresponding to a Byzantine node i may be arbitrary and need not follow (8).
- *Peer-to-peer architecture*: In the local computation phase, each honest node i computes $\theta_{t+1/2}^{(i)} = \theta_t^{(i)} - \gamma_t^{(i)} m_t^{(i)}$—that is, it uses the momentum of its gradient to compute the partial update. The communication phase remains the same as in Section 3.2.2. The global update phase is the same as that in Section 5.1.

The convergence of the preceding adaptation of D-SGD in the server-based architecture has been analyzed for almost all the aforementioned aggregation rules (except CGE[6]), namely (Multi-)Krum, GM, MDA, CwMed, MeaMed, CwTM, and CC, in the work of Farhadkhani et al. [73] and Karimireddy et al. [109]. The convergence of the peer-to-peer variant has only been analyzed for an aggregation rule called *Nearest-Neighbor Averaging (NNA)* [72].

5.2.2 D-SGD with Dynamic Sampling and Robust Aggregation. The dynamic sampling is incorporated in the local computation phase. Specifically, in the local computations phase of each iteration t, each honest node i computes point-wise gradients for a mini-batch of data points, represented by set $S_t^{(i)}$, sampled from its local distribution $\mathcal{D}_i$. The size of the mini-batch is proportional to the number of iterations (i.e., $|S_t^{(i)}| = t$). Specifically, for each honest node i, the stochastic gradient $g_t^{(i)}$ is redefined as follows:

$$g_t^{(i)} = \frac{1}{\left|S_t^{(i)}\right|} \sum_{x \in S_t^{(i)}} \nabla q\left(\theta_t^{(i)}, x\right). \tag{9}$$

The rest of the algorithm is identical to robust D-SGD described in Section 5.1. The preceding adaptation, a.k.a. *LEARN*, has only been analyzed for two aggregation rules, namely MDA and CwTM [65].

5.2.3 Breakdown, Robustness, and Gradient Complexity. The use of local variance reduction (either through momentum or techniques like dynamic sampling and VR-MARINA) in D-SGD eliminates the need for non-classical assumptions for proving Byzantine resilience, as shown elsewhere [73, 85]. We make the following key observations based on the results presented in other works [65, 72, 73, 85, 108]:

- All the aforementioned adaptations of D-SGD with momentum obtain optimal breakdown point of 1/2 without a significant increase in the gradient complexity. This holds with the exception of CC [109] and NNA [72] that have breakdown points of 1/10 and 1/5, respectively.
- CC and NNA are the only aggregation rules that provably provide optimal robustness (but in expectation) [72, 108]. In other words, the expected asymptotic error of D-SGD with

[6]The CGE aggregation rule does not satisfy the basic property of *sanity* (defined in Appendix A), which makes it rather vulnerable to Byzantine perturbations even when the inputs of all honest nodes are identical [73].

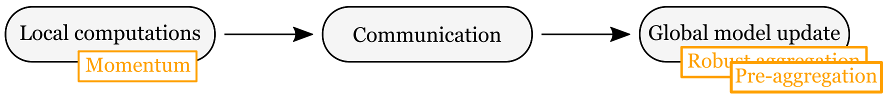

Fig. 8. D-SGD with a robust aggregation and a pre-aggregation at the update phase and a momentum scheme at the local phase.

momentum and either CC or NNA matches the lower bound discussed in Section 4.3 under heterogeneity. However, CC is an iterative algorithm that can only be shown to be *concentrated* around the true average when we have *a priori* knowledge on the variance of the honest inputs, and have an initial guess on the average of the honest vectors with *known* bounded error [109].

- Both CC and NNA have gradient complexity of $O(\frac{f+1}{n} \frac{1}{\varepsilon^2})$ for obtaining (f, ε)-Byzantine resilience, under homogeneity (i.e., when the local data distributions for honest nodes are identical), which is the best known gradient complexity for a Byzantine resilient machine learning solution (in the non-convex case). However, the gradient complexities for (Multi-)Krum, GM, MDA, CwMed, MeaMed, and CwTM, as analyzed in the work of Allouah et al. [11] and Farhadkhani et al. [73], is in $O((\frac{f+1}{n} + \zeta) \frac{1}{\varepsilon^2})$, where the specific value of $\zeta > 0$ depends upon the aggregation rule.
- Although MDA and CwTM can guarantee optimal robustness under dynamic sampling (i.e., LEARN) [65], the resulting gradient complexity due to dynamic sampling is orders of magnitude higher in comparison to using momentum, which has comparable complexity to D-SGD. Furthermore, the convergence rate of LEARN is not known, which makes the gradient complexity hard to evaluate.

5.3 D-SGD with Momentum and Two-Step Aggregation

The augmentation of D-SGD with momentum and robust aggregation rules shows promising guarantees on both the breakdown point and gradient complexity. To address the sub-optimality of this algorithm in terms of robustness, two meta schemes called *bucketing* and *Nearest-Neighbor Mixing (NNM)* were proposed by Karimireddy et al. [108] and Allouah et al. [11], respectively. We call these schemes *pre-aggregation*, as they intervene prior to the aggregation rule, overall yielding a two-step aggregation. The algorithm combining a two-step aggregation with momentum, illustrated in Figure 8, follows the same steps as the method described in Section 5.2.1 but replaces F with $F \circ \text{PRE-AGG}$, where $\circ$ is the composition operator between two functions and is the pre-aggregation at hand, be it bucketing or NNM. We formally detail the procedure for both these pre-aggregations next.

5.3.1 Pre-Aggregation by Bucketing.

Bucketing is a pre-aggregation method that averages random subsets of inputs prior to feeding them to an aggregation rule F during the global model update phase. Given a collection of n vectors $x_1, \ldots, x_n \in \mathbb{R}^d$, bucketing first randomly shuffles the inputs. Formally, let us consider a random permutation $\tau : \{1, \ldots, n\} \to \{1, \ldots, n\}$, then bucketing maps $\{x_1, \ldots, x_n\}$ to an ordered tuple $(x_{\tau(1)}, \ldots, x_{\tau(n)})$. Then, the scheme breaks the tuple $(x_{\tau(1)}, \ldots, x_{\tau(n)})$ into *buckets* of size $s \geq 1$ (or less[7]), denoted by $\text{bucket}_1, \ldots, \text{bucket}_{\lceil n/s \rceil}$. In other words, for each $i \in \{1, \ldots, \lceil n/s \rceil\}$, $\text{bucket}_i = \{\tau((i-1)s+1), \ldots, \tau(\min\{is, n\})\}$. Finally, bucketing computes the average of the vectors in each bucket. Specifically, the methods outputs

[7]If n is a multiple of s, each bucket will have s vectors. Otherwise, we will have one bucket of size r, where $r = n$ modulo s.

$\textsc{Buck}(x_1, \ldots, x_n) := y_1, \ldots, y_{\lceil n/s \rceil}$, where for each $i \in \{1, \ldots, \lceil n/s \rceil\}$,

$$y_i = \frac{1}{s} \sum_{j \in \text{bucket}_i} x_j.$$

5.3.2 Pre-Aggregation by NNM. Unlike bucketing, NNM is a deterministic pre-aggregation method. Given a collection of n vectors $x_1, \ldots, x_n \in \mathbb{R}^d$, NNM replaces every vector with the average of its $n - f$ nearest neighbors (including itself). Formally, the method outputs $\text{NNM}(x_1, \ldots, x_n) := (y_1, \ldots, y_n)$ where for each $i \in [n]$,

$$y_i = \frac{1}{n-f} \sum_{j=1}^{n-f} x_{i:j} \ ,$$

and $x_{i:j}$ is the j-th nearest vector (in terms of the Euclidean distance) to x_i in $(x_1, \ldots, x_n)$.

5.3.3 Breakdown, Robustness, and Gradient Complexity. Although pre-aggregation enhances Byzantine resilience, the benefits do not come for free. We describe the merits and limitations of bucketing and NNM as follows:

- These schemes have been shown to improve the Byzantine resilience of four aggregation rules, namely Krum, GM, CwMed, and CwTM [11, 108]. Specifically, using either NNM or bucketing in conjugation with any of these aggregation rules yields optimal robustness to D-SGD with momentum. Furthermore, we also obtain an improved gradient complexity $O(\frac{f+1}{n} \frac{1}{\varepsilon^2})$ for achieving (f, ε)-Byzantine resilience (under homogeneity).
- However, both bucketing and NNM have some limitations. In bucketing, since we simply average the input vectors, there are potentially f buckets out of s that can get corrupted by Byzantine nodes.[8] Accordingly, using bucketing with $s > 1$ reduces the breakdown point of the overall algorithm. This is especially true in the case of Krum, where the breakdown point deteriorates to $1/4$ instead of $1/2$ (shown in the work of Farhadkhani et al. [73]). This limitation can be circumvented using NNM as shown in the work of Allouah et al. [11]. In fact, NNM offers an almost-optimal breakdown point of $\frac{1+\eta}{2}$ for any given $\eta > 0$. However, this comes at the cost of computational efficiency. Indeed, the computational complexity of NNM is in $O(dn^2)$. Although this cost does not prevent scalability (defined in Appendix A), it can hinder the usage of NNM for large-scale distributed learning tasks.

Remark 3. Note that the convergence result from the work of Karimireddy et al. [108] incorporates randomness due to the shuffling operation used for bucketing. This source of randomness cannot be overlooked even when the honest nodes compute the exact true gradients of their local loss functions. This might amplify the uncertainty in the computations, hence resulting in weaker probabilistic convergence guarantees than other solutions reaching optimal robustness, such as CC, NNA, and NNM.

6 CONCLUSION AND DISCUSSION

We have introduced a new perspective on the problem of Byzantine machine learning to standardize the evaluation of different solutions. Our approach presents a clear outlook on the merits and limitations of prominent existing solutions. Our introduced system of evaluation is generic and can be utilized for evaluating other methods (e.g., see [37, 38, 40, 43, 52, 61, 79, 94, 130, 140, 159,

[8]As the average is arbitrarily manipulable, the Byzantine nodes can get full control over their buckets, effectively rendering the buckets Byzantine.

199, 204]) that were not explicitly considered in the article. Finally, our systematization also allowed us to identify promising upcoming trends in Byzantine machine learning. Next, we present the primary takeaway from our survey.

6.1 Tensions between Breakdown Point, Robustness, and Gradient Complexity

An ideal solution to Byzantine machine learning should attain optimality in all the three fronts: (i) tolerate up to half of the Byzantine nodes, (ii) ensure a training loss that matches the established lower bound, and (iii) have minimal overhead on gradient computations. However, existing schemes seem to only ensure optimality on at most two out of the three metrics. To put it succinctly, existing works exhibit tensions between breakdown point, robustness, and gradient complexity. Hence, an important question that remains to be addressed is the following.

Does there exist a fundamental tradeoff that hinders the simultaneous optimization of breakdown point, robustness, and gradient complexity in Byzantine machine learning?

Next, we discuss some other future research directions that we believe to be of importance to align Byzantine machine learning solutions with the current practices in machine learning at large.

6.2 Asynchronous Systems

As the number of nodes grows, we must expect larger speed differences between the fastest and slowest machines in the system. Furthermore, some nodes may suddenly stall (e.g., because of concurrent applications or a jammed network [36]). It is then critical that the entire system does not get paralyzed, whenever a handful of nodes fail to respond at their regular pace. This has prompted the design of distributed machine learning algorithms that can tolerate asynchronous environments—that is, systems where the nodes may undergo unknown communication delays [7, 53, 125, 160]. In addition to causing potential system paralysis, asynchrony is known to make Byzantine resilience much more difficult, as indicated by the famous impossibility of asynchronous fault-tolerant consensus [78]. However, most existing works in Byzantine machine learning only consider synchronous communication, with a few exceptions [51, 65, 72]. To fully harness the potential of distributed learning in real-world applications, we strongly believe that it is imperative to design solutions for Byzantine machine learning that account for asynchrony.

6.3 Communication Efficiency

Another crucial aspect of distributed algorithms, which have to be tackled before deploying them in real-world applications, is their communication efficiency. Indeed, the communication cost of sending the gradient often becomes the bottleneck of the learning procedure in distributed environments [120, 121]. To address this problem, several recent works have focused on reducing the communication cost of distributed learning algorithms using gradient quantization, compression and sparsification [8, 9, 14, 59, 105, 172, 173, 200, 201, 216]. These methods have been shown to significantly improve the communication cost of standard D-SGD, which is particularly attractive for large-scale distributed machine learning. However, in a large-scale system, it is arguably inevitable to encounter faulty devices that may deviate from the prescribed algorithm. While some recent work have been trying to adapt these solution to integrate the possibility for Byzantine nodes [21, 81, 83, 207], the literature remains quite scarce on this topic. We believe that along gradient complexity, communication efficiency should be a central metric to consider for deployment of Byzantine machine learning at scale.

6.4 Sparse Communication Topology

An emerging interest in peer-to-peer architecture for distributed machine learning is scalability of the system, by using a sparse communication network [32, 113, 126]. However, existing Byzantine machine learning solutions often rely on dense (or even complete) communication topology [65, 72, 97, 114]. Such requirements undermine the applicability of these methods in practice, which has motivated recent works on obtaining Byzantine resilience in distributed machine learning over incomplete sparse network [69, 100]. However, these methods have very small breakdown points, and they rely on non-classical assumptions on data distributions and loss functions. Designing Byzantine machine learning solutions over sparse communication networks with optimal breakdown point, robustness, and gradient complexity remains an interesting open problem.

6.5 Circumventing the Lower Bounds in Practice

Often the breakdown point of 1/2 is considered quite restrictive in practical settings. There have been some works where the server is assumed to have additional knowledge on data, in which case the breakdown point can be much higher than 1/2 [39, 161, 208, 209, 213]. This is an interesting approach that warrants further consideration in the future. Moreover, we can also utilize some form of randomized checks during the learning procedure to avoid the use of robust aggregation rules in every iteration [94]. By doing so, not only can we economize the computation in the global model update phase, but we may also improve upon the gradient complexity of the algorithm. Moreover, by incorporating the suspicion-based outlier detection schemes (e.g., [23, 156]), along with a robust aggregation rule, we may also be able to circumvent the lower bound on robustness.

6.6 Applicability to Adaptive Optimizers

The machine learning community is becoming increasingly interested in variants of SGD, such as AdaGrad [62] and Adam [112], where the learning rate adapts over the course of the learning procedure for improved accuracy under sparse sampling of data [54]. The adaptive methods have become a standard in training of large neural networks owing to their superior performance compared to standard SGD. However, at this point, the Byzantine machine learning community predominantly focuses on first-order update rules with static (i.e., non-adaptive) learning rates, with the exception of some works [10, 215] that have considered second-order update schemes. For Byzantine machine learning solutions to be compatible with contemporary practices in modern machine learning, it is important to design and study their extension to adaptive gradient descent methods in the near future. More generally, investigating or addressing the impact of Byzantine resilience on the generalizability of the learning algorithm remains an interesting open problem.

6.7 Synthesis with Privacy

It is only a matter of time before machine learning solutions become standard in sensitive public-domain applications such as medicine and banking. For this reason, privacy preservation of training data becomes necessary, both to address growing concerns about the processing of personal data and to comply with new legislation (e.g., GDPR) on the use of AI-based technologies. Common privacy protection schemes in distributed machine learning include *differential privacy* [102, 145] and *homomorphic encryption* [86, 217]. These schemes aim to protect the privacy of nodes' local data against the server and other nodes in the system. Although these schemes have been widely studied on their own, their synthesis with Byzantine robustness remains poorly understood. Recent works demonstrate the difficulty in ensuring differential privacy and robustness simultaneously [12, 89, 122, 132, 224]. Designing a scalable and computationally efficient scheme that could provide privacy and robustness simultaneously is an important active area of research.

ACKNOWLEDGMENTS

The authors are thankful to Sasha Voitovych for proofreading the initial version of this article, as well to Youssef Allouah, Sadegh Farhadkhani and John Stephan for fruitful discussions. The authors would also like to thank the anonymous reviewers and the action editor for their constructive comments that helped improve the overall presentation of the article.

APPENDIX

A DETAILS AND CLASSIFICATION OF AGGREGATION RULES

In this appendix, we present details on the aggregation rules mentioned in the main body of the article and analyze their merits through three criteria (we introduce in the following): sanity, concentration, and scalability. The characterization of presented aggregation rules based on these notions is summarized in Figure 9. Throughout the appendix, for a positive integer n, we denote the set $\{1, \ldots, n\}$ by $[n]$.

A.1 Sanity, Concentration, and Scalability

We introduce elementary criteria that standardize the approach for verifying whether an aggregation rule is a valid candidate for Byzantine resilient machine learning. These criteria are simple and agnostic to the learning task—that is, they depend only on the aggregation rule and not on the distributed learning algorithm. The first two, namely *sanity* and *concentration*, are based on a minimal guarantee of robustness for the aggregation. Essentially, they ensure that a Byzantine node cannot arbitrarily manipulate the method, and that it respects the majority consensus (when it exists). The third one, namely *scalability*, is related to practical considerations. We detail these criteria next.

A.1.1 Sanity. This condition ensures that if the honest nodes have identical inputs, then the robust aggregation outputs their input value. Formally, sanity is defined as follows.

Definition 5 (Sanity). Let $f < n$. For any $y \in \mathbb{R}^d$, we denote by $\mathcal{S}_y$ the subset of $\mathbb{R}^{d\times n}$ for which at least $n - f$ elements are equal to y—that is, $\mathcal{S}_y := \{(x_1, \ldots, x_n) \in \mathbb{R}^{d\times n} \mid \exists I \subset [n] \text{ s.t. } |I| \geq n - f \wedge x_i = y, \forall i \in I\}$. An aggregation rule $F \colon \mathbb{R}^{d\times n} \to \mathbb{R}^d$ is said to satisfy *sanity* if for any $y \in \mathbb{R}^d$, the following holds true:

$$F(x_1, \ldots, x_n) = y\,, \quad \forall (x_1, \ldots, x_n) \in \mathcal{S}_y.$$

Sanity, although simple, is an important property for an aggregation rule. Indeed, although Byzantine machine learning usually considers complex tasks involving *multiple stochastic oracles*, sanity ensures that when the gradients for honest nodes are generated by a *single deterministic oracle*, the aggregation rule mimics the majority voting scheme, which is known to be optimal in this context [131].

A.1.2 Concentration. This criterion concerns the ability of the aggregation rule to estimate the true average of honest nodes' inputs. However, since the identity of the honest nodes is *a priori* unknown, the aggregation rule must guarantee a more sophisticated property. Essentially, concentration states that for any subset of inputs of size $n - f$, the output of the aggregation rule is in the proximity of the average of these inputs, where the proximity is controlled by a function $\Psi : \mathbb{R}^{d\times(n-f)} \to \mathbb{R}_+$. Formally, concentration is defined as follows.

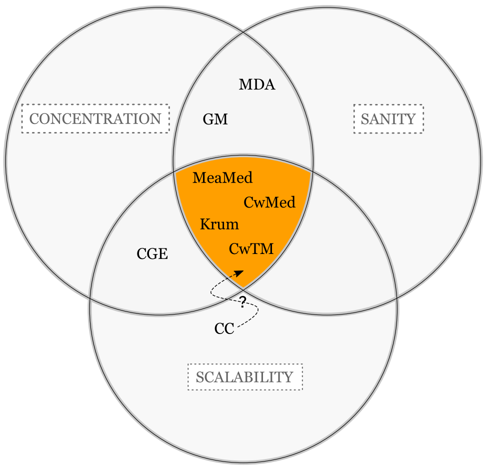

Fig. 9. Characterization of prominent aggregation rules, based on the notions of sanity, concentration, and scalability.

Definition 6 (Concentration). For $f < n$, an aggregation rule $F\colon \mathbb{R}^{d \times n} \to \mathbb{R}^d$ is said to satisfy *concentration* if there exists a real valued function $\Psi\colon \mathbb{R}^{d \times (n-f)} \to \mathbb{R}_+$ such that for any collection of n random vectors $x_1, \ldots, x_n$, and any set $I \subseteq \{1, \ldots, n\}$ of size $n - f$,

$$\mathbb{E}\left[\|F(x_1, \ldots, x_n) - \bar{x}_I\|^2\right] \leq \mathbb{E}\left[\Psi\left(x_i; i \in I\right)\right],$$

where $\bar{x}_I := \frac{1}{n-f}\sum_{i \in I} x_i$ and the expectation $\mathbb{E}[\cdot]$ is taken over the randomness of the inputs.

Note that Definition 6 is rather general as it does not specify the nature of the function Ψ. In principle, this function should be a measure of the dispersion of the inputs so that $F(x_1, \ldots, x_n)$ is a good approximation of the average of a subset of inputs with small dispersion. In recent years, several conditions have been proposed for proving Byzantine resilience of a learning procedure that can relate to the notion of concentration, notably (δ, c)*-robust averaging* [109], C*-averaging agreement* [65], and (f, λ)*-resilient averaging* [73]. Indeed, an aggregation rule that satisfies one of these notions automatically satisfies concentration.

A.1.3 Scalability. The third criterion we consider is related to the computational efficiency of the aggregation rule. Designing an aggregation rule that works perfectly in theory but is hard to implement in practice is not desirable in machine learning, especially for large-scale and high-stack applications. Accordingly, we propose to consider aggregation rules that have reasonable time complexity and can scale well with system size. Formally, we define the property of *scalability* as follows.

Definition 7 (Scalability). For $f < n$, an aggregation rule $F\colon \mathbb{R}^{d \times n} \to \mathbb{R}^d$ is said to satisfy scalability if its worst-case time complexity is polynomial n (i.e., also in f) and at most linear in d.

We now present some prominent aggregation rules and characterize them using the three aforementioned criteria. In doing so, we divide them in three categories: *coordinate-wise* (CwMed, CwTM, and MeaMed), *geometric centers* (Krum, GM, and MDA), and *norm-based filtering* (CGE and CC) methods.

A.2 Coordinate-wise Aggregation Rules

These aggregation rules operate on a coordinate-wise basis. Prominent coordinate-wise aggregation rules include CwMed, CwTM, and MeaMed.

A.2.1 Coordinate-wise Median. Let $x \in \mathbb{R}^d$, and we denote by $[x]_k$ the k-th coordinate of x. The first candidate to replace the average is naturally the median of the vectors. However, as we discuss later in this section, the median is non-trivial to compute in high-dimensional settings. Thus, the simplest solution is to compute the median coordinate-wise. Formally, for input vectors $x_1, \ldots, x_n$, their coordinate-wise median, denoted by CwMed$(x_1, \ldots, x_n)$, is defined to be a vector whose k-th coordinate, for all $k \in [d]$, is defined to be

$$[\text{CwMed}\,(x_1, \ldots, x_n)]_k \coloneqq \text{Median}\,([x_1]_k, \ldots [x_n]_k)\,. \tag{10}$$

CwMed clearly ensures sanity and has been demonstrated to satisfy (λ, f)-resilient averaging, a specific concentration criterion [73]. Furthermore, computing the coordinate-wise median is very time efficient. Indeed, it suffices to compute a median for each coordinate, which only demands a linear amount of computations [27]. Accordingly, CwMed has an overall time complexity of $O(nd)$, which matches the time complexity of computing the average of d-dimensional vectors and satisfies scalability.

A.2.2 Coordinate-wise Trimmed Mean. Given a collection of n input vectors $x_1, \ldots, x_n \in \mathbb{R}^d$, we denote by τ_k the permutation on $[n]$ that sorts the k-coordinate of the input vectors in non-decreasing order—that is, $[x_{\tau_k(1)}]_k \leq [x_{\tau_k(2)}]_k \leq \cdots \leq [x_{\tau_k(n)}]_k$. Then, the CwTM of $x_1, \ldots, x_n$, denoted by CwTM$(x_1, \ldots, x_n)$, is a vector in $\mathbb{R}^d$ whose k-th coordinate is defined as follows:

$$[\text{CwTM}(x_1, \ldots, x_n)]_k \coloneqq \frac{1}{n-2f} \sum_{j \in [f+1, n-f]} [x_{\tau_k(j)}]_k.$$

CwTM has been shown to ensure both (λ, f)-resilient averaging [73] and C-averaging agreement [65]. Accordingly, it also ensures concentration and sanity [73]. Furthermore, the time complexity of sorting vectors in each dimension is in $O(dn\log(n))$ and averaging the selected value in each dimension is in $O(nd)$. Overall, the time complexity of CwTM is in $O(dn\log(n))$, which is only a $log(n)$ factor away from complexity of computing the average. Thus, CwTM satisfies scalability.

A.2.3 Mean of Medians. Given a collection of n input vectors $x_1, \ldots, x_n$, MeaMed computes the average of the $n - f$ closest elements to the median in each dimension. Specifically, for each $k \in [d]$, $m \in [n]$, let $i_{m;k}$ be the index of the input vector with k-th coordinate that is the m-th closest to Median$([x_1]_k, \ldots, [x_n]_k)$. Let C_k be the set of $n - f$ indices defined as

$$C_k = \{i_{1;k}, \ldots, i_{n-f;k}\}.$$

Then we have

$$[\text{MeaMed}(x_1, \ldots, x_n)]_k = \frac{1}{n-f} \sum_{i \in C_k} [x_i]_k,$$

where MeaMed$(x_1, \ldots, x_n)$ denotes the output of the aggregation rule.

Similar to CwMed and CwTM, MeaMed satisfies concentration and sanity through (λ, f)-resilient averaging [73]. Furthermore, to compute the median as well as the $n - f$ closest vector to the median in each dimension, it suffices to sort each dimension hence incurring a complexity

$O(nd\log(n))$. Accordingly, the complexity of MeaMed is the same as the complexity of CwTM. Hence, MeaMed satisfies scalability.

A.3 Geometric Centers

Beyond coordinate-wise computations, several other aggregation rules have been proposed that aim at providing a more geometric solution. These techniques try to find a vector that would be a good representative of the "center" of the input vectors. These include Krum, GM, and MDA.

A.3.1 Krum and Multi-Krum. Given a collection of input vectors $x_1, \dots, x_n$, Krum outputs a vector $x \in \{x_1, \dots, x_n\}$ that is closest to its neighbors upon discarding the f farthest ones. Specifically, for each $i \in [n]$ and $j \in [n-1]$, let $i_j \in [n] \setminus \{i\}$ be the index of the j-th closest input vector from x_i—that is, we have $\|x_i - x_{i_1}\| \leq \cdots \leq \|x_i - x_{i_{n-1}}\|$. Let C_i be the set of $n - f - 1$ closest vectors to x_i—that is,

$$C_i = \{i_1, \dots, i_{n-f-1}\}.$$

Finally, Krum outputs the input vector that has the smallest cumulative distance to their $n - f$ closest neighbors—that is,

$$\text{Krum}\,(x_1, \dots, x_n) = x_{i^*} \quad \text{where} \quad i^* \in \underset{i \in [n]}{\operatorname{argmin}} \sum_{j \in C_i} \|x_i - x_j\|^2. \tag{11}$$

The original scheme was further enhanced under the name Multi-Krum$_q$. This aggregation rule outputs the average of the q input vectors with the smallest cumulative distances—that is,

$$\text{Multi-Krum}_q\,(x_1, \dots, x_n) = \frac{1}{q} \sum_{i \in M(q)} x_i,$$

where $M(q)$ is the set of q vectors with the smallest cumulative distances to their neighbors. Note that Krum is essentially Multi-Krum$_q$ for $q = 1$.

Krum and Multi-Krum satisfy both sanity and concentration, as per Farhadkhani et al. [73]. Furthermore, the time complexity of computing all pairwise distances between the vectors is in $O(dn^2)$ and the complexity of computing the cumulative distances for each input is in $O(n^2\log(n))$. Finally, the argmin computation is in $O(n)$. Overall, Krum's computational complexity is $O(n^2(d + \log(n)))$. Hence, Krum belongs to the class of scalable aggregation rules, although it is much more expensive than coordinate-wise methods. The same holds true for Multi-Krum.

A.3.2 Geometric Median. For input vectors $x_1, \dots, x_n$, their GM, denoted by $\text{GM}(x_1, \dots, x_n)$, is defined to be a vector that minimizes the sum of the distances to these vectors. Specifically, we have

$$\text{GM}(x_1, \dots, x_n) \in \underset{z \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{n} \|z - x_i\|\,.$$

GM ensures both sanity and concentration, and has been analyzed under several frameworks, notably through (λ, f)-resilient averaging [73] and C-averaging agreement [65]. However, in general, there exists no closed form for GM. Accordingly, it does not satisfy scalability in theory. Existing methods implementing an approximation of GM (e.g., [35, 46, 50, 148, 151] and references therein) are based on iterative algorithms. Yet, most of these methods require expensive computations—for example, determining eigenvalues and eigenvectors of $d \times d$ matrices [50] in each iteration.

A.3.3 Minimum Diameter Averaging. Given a set of n vectors $x_1, \ldots, x_n \in \mathbb{R}^d$, the MDA algorithm first chooses a set S^* of cardinality $n - f$ with the smallest *diameter*—that is,

$$S^* \in \underset{\substack{S \subset \{1,\ldots,n\} \\ |S|=n-f}}{\operatorname{argmin}} \left\{ \max_{i,j \in S} \|x_i - x_j\| \right\}. \tag{12}$$

Then the MDA outputs the average of the inputs in set S^*. Specifically,

$$\mathrm{MDA}(x_1, \ldots, x_n) \coloneqq \frac{1}{n-f} \sum_{i \in S^*} x_i. \tag{13}$$

Similar to GM, MDA ensures both sanity and concentration through both (λ, f)-resilient averaging [73] and C-averaging agreement [65]. However, it does not satisfy scalability. Indeed, the time complexity of computing all pairwise distances between the vectors is in $O(dn^2)$. Furthermore, given the table of pairwise distances, the time complexity of finding the set S^* with the smallest diameter is in $O\left(\binom{n}{f}\right)$. Overall, the worst-case time complexity of MDA is in $O\left(\binom{n}{f} + dn^2\right)$, which can be prohibitive when f is comparable to n.

A.4 Norm-Based Filtering Methods

Finally, we discuss the two recent aggregation rules (CC and CGE) that have been designed in an attempt to filter or normalize the gradients as per their Euclidean norms.

A.4.1 Centered Clipping. Given the input vectors $x_1, \ldots, x_n \in \mathbb{R}^d$, upon choosing a *clipping parameter* $c_\tau \geq 0$, we compute a sequence of vectors $v_0, \ldots, v_M$ in $\mathbb{R}^d$ such that for all $m \in [M]$,

$$v_m \leftarrow v_{m-1} + \frac{1}{n} \sum_{i \in [n]} (x_i - v_{m-1}) \min \left\{ 1, \frac{c_\tau}{\|x_i - v_{m-1}\|} \right\},$$

where v_0 may be chosen arbitrary. Then, $\mathrm{CC}(x_1, \ldots, x_n) = v_M$.

CC is an iterative algorithm, hence it can be considered scalable if M is small enough. The value of M that enable convergence of the procedure toward a satisfactory point, however, depends highly upon the inputs and the parameters used to compute the aggregation rule. As shown in the work of Karimireddy et al. [109], by setting specific values for parameters c_τ and M, CC can satisfy the condition of (c, δ)-robust averaging. However, this relies on extra information that is often not accessible in practice. Specifically, the values for parameters c_τ and M depend on the maximal variance of the honest gradients and on a bound on the initial estimate error $\mathbb{E}\left[\|\overline{x_{\mathcal{H}}} - v_0\|^2\right]$. Analyzing CC under standard assumptions and without any extra information remains an open question. Hence, it remains unclear whether it can satisfy sanity and/or concentration.

A.4.2 Comparative Gradient Elimination. For input vectors $x_1, \ldots, x_n$, let τ denote a permutation on $[n]$ that sorts the input vectors based on their norm and in non-decreasing order—that is, $\|x_{\tau(1)}\| \leq \|x_{\tau(2)}\| \leq \cdots \leq \|x_{\tau(n)}\|$. CGE outputs the average of the $n - f$ vectors with smallest norm [92]—that is,

$$\mathrm{CGE}(x_1, \ldots, x_n) = \frac{1}{n-f} \sum_{i=1}^{n-f} x_{\tau(i)}.$$

CGE only needs to compute the norm of each input and to sort this values before averaging them. Hence, similar to CwTM, this scheme scales as its worst-case complexity is in $O(n(d + \log(n)))$. Furthermore, CGE can easily be shown to satisfy concentration by setting $\Psi(x_i : i \in I) \coloneqq {}^{f^2}\!/_{(n-f)^2} \max_{i \in I} \|x_i\|^2$. However, CGE does not satisfy sanity. Specifically, consider the case when $n - f$ input vectors are identically equal to x^* with $\|x^*\| > 0$. If the remaining f vectors have smaller norms than $\|x^*\|$, then the output of CGE is not equal to x^*, violating sanity as per Definition 5.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv:1603.04467* (2015).

[2] Tarek Abdelzaher, Nora Ayanian, Tamer Basar, Suhas Diggavi, Jana Diesner, Deepak Ganesan, Ramesh Govindan, Susmit Jha, Tancrede Lepoint, Benjamin Marlin, Klara Nahrstedt, David Nicol, Raj Rajkumar, Stephen Russell, Sanjit Seshia, Fei Sha, Prashant Shenoy, Mani Srivastava, Gaurav Sukhatme, Ananthram Swami, Paulo Tabuada, Don Towsley, Nitin Vaidya, and Venu Veeravalli. 2018. Toward an Internet of Battlefield Things: A resilience perspective. *Computer* 51, 11 (2018), 24–36.

[3] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. 2012. *Learning from Data*. Vol. 4. AMLBook, New York, NY.

[4] Anish Acharya, Abolfazl Hashemi, Prateek Jain, Sujay Sanghavi, Inderjit S. Dhillon, and Ufuk Topcu. 2022. Robust training in high dimensions via block coordinate geometric median descent. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (Eds.), Vol. 151. PMLR, 11145–11168. https://proceedings.mlr.press/v151/acharya22a.html

[5] Mostofa Ahsan, Kendall E. Nygard, Rahul Gomes, Md. Minhaz Chowdhury, Nafiz Rifat, and Jayden F. Connolly. 2022. Cybersecurity threats and their mitigation approaches using machine learning—A review. *Journal of Cybersecurity and Privacy* 2, 3 (2022), 527–555.

[6] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. 2018. Byzantine stochastic gradient descent. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 4618–4628.

[7] Dan Alistarh, Christopher De Sa, and Nikola Konstantinov. 2018. The convergence of stochastic gradient descent in asynchronous shared memory. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*. 169–178.

[8] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. 2017. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. 1707–1718.

[9] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. 2018. The convergence of sparsified gradient methods. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*. 5977–5987.

[10] Zeyuan Allen-Zhu, Faeze Ebrahimianghazani, Jerry Li, and Dan Alistarh. 2020. Byzantine-resilient non-convex stochastic gradient descent. In *Proceedings of the International Conference on Learning Representations*.

[11] Youssef Allouah, Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Rafaël Pinot, and John Stephan. 2023. Fixing by mixing: A recipe for optimal Byzantine ML under heterogeneity. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*. 1232–1300.

[12] Youssef Allouah, Rachid Guerraoui, Nirupam Gupta, Rafaël Pinot, and John Stephan. 2023. On the privacy-robustness-utility trilemma in distributed learning. In *Proceedings of the International Conference on Machine Learning (ICML'23)*.

[13] Mohammad Abu Alsheikh, Shaowei Lin, Dusit Niyato, and Hwee Pink Tan. 2014. Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *IEEE Communication Surveys & Tutorials* 16, 4 (2014), 1996–2018. https://doi.org/10.1109/COMST.2014.2320099

[14] By Mahmoud Assran, Arda Aytekin, Hamid Reza Feyzmahdavian, Mikael Johansson, and Michael G. Rabbat. 2020. Advances in asynchronous parallel and distributed optimization. *Proceedings of the IEEE* 108, 11 (2020), 2013–2031. https://doi.org/10.1109/JPROC.2020.3026619

[15] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning (ICML'18)*. https://arxiv.org/abs/1802.00420

[16] Hagit Attiya and Jennifer Welch. 2004. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. Vol. 19. John Wiley & Sons.

[17] Yixin Bao, Yanghua Peng, Chuan Wu, and Zongpeng Li. 2018. Online job scheduling in distributed machine learning clusters. In *Proceedings of the IEEE Conference on Computer Communications (IEEE INFOCOM'18)*. IEEE, Los Alamitos, CA, 495–503.

[18] Moran Baruch, Gilad Baruch, and Yoav Goldberg. 2019. A little is enough: Circumventing defenses for distributed learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS'19)*. 8635–8645.

[19] Bernard W. Bell. 2021. Replacing bureaucrats with automated sorcerers? *Daedalus* 150, 3 (2021), 89–103.

[20] Tal Ben-Nun and Torsten Hoefler. 2019. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Computing Surveys* 52, 4 (2019), 1–43.

[21] Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. 2019. SignSGD with majority vote is communication efficient and fault tolerant. *arXiv:cs.DC/1810.05291* (2019).

[22] Dimitri Bertsekas and John Tsitsiklis. 2015. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific.

[23] Kush Bhatia, Prateek Jain, and Purushottam Kar. 2015. Robust regression via hard thresholding. In *Proceedings of the 28th International Conference on Neural Information Processing Systems—Volume 1 (NIPS'15)*. MIT Press, Cambridge, MA, 721–729.

[24] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 387–402.

[25] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning (ICML'12)*. 1467–1474.

[26] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. 118–128.

[27] Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan. 1972. Linear time bounds for median computations. In *Proceedings of the 4th Annual ACM Symposium on Theory of Computing (STOC'72)*. ACM, New York, NY, 119–124. https://doi.org/10.1145/800152.804904

[28] Léon Bottou. 1999. On-line learning and stochastic approximations. In *On-Line Learning in Neural Networks*, David Saad (Ed.). Cambridge University Press, 9–42. https://doi.org/10.1017/CBO9780511569920.003

[29] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. 2018. Optimization methods for large-scale machine learning. *SIAM Review* 60, 2 (2018), 223–311.

[30] Djamila Bouhata and Hamouma Moumen. 2022. Byzantine fault tolerance in distributed machine learning: A survey. *arXiv preprint arXiv:2205.02572* (2022).

[31] Amine Boussetta, El Mahdi El Mhamdi, Rachid Guerraoui, Alexandre Maurer, and Sébastien Rouault. 2021. AKSEL: Fast Byzantine SGD. In *Proceedings of the 24th International Conference on Principles of Distributed Systems (OPODIS'20)*.

[32] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Now Foundations and Trends.

[33] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

[34] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS'20)*. 1877–1901.

[35] Mihai Bundefineddoiu, Sariel Har-Peled, and Piotr Indyk. 2002. Approximate clustering via core-sets. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC'02)*. ACM, New York, NY, 250–257. https://doi.org/10.1145/509907.509947

[36] Christian Cachin, Rachid Guerraoui, and Luís Rodrigues. 2011. *Introduction to Reliable and Secure Distributed Programming*. Springer Science & Business Media.

[37] Di Cao, Shan Chang, Zhijian Lin, Guohua Liu, and Donghong Sun. 2019. Understanding distributed poisoning attack in federated learning. In *Proceedings of the 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS'29)*. IEEE, Los Alamitos, CA, 233–239.

[38] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. 2021. FLTrust: Byzantine-robust federated learning via trust bootstrapping. In *Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS'21)*.

[39] Xinyang Cao and Lifeng Lai. 2019. Distributed gradient descent algorithm robust to an arbitrary number of Byzantine attackers. *IEEE Transactions on Signal Processing* 67, 22 (2019), 5850–5864.

[40] Xiaoyu Cao, Zaixi Zhang, Jinyuan Jia, and Neil Zhenqiang Gong. 2022. FLCert: Provably secure federated learning against poisoning attacks. *IEEE Transactions on Information Forensics and Security* 17 (2022), 3691–3705.

[41] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP'17)*. IEEE, Los Alamitos, CA, 39–57.

[42] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. 2017. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 47–60.

[43] Lingjiao Chen, Hongyi Wang, Zachary B. Charles, and Dimitris S. Papailiopoulos. 2018. DRACO: Byzantine-resilient distributed training via redundant gradients. In *Proceedings of the 35th International Conference on Machine Learning*, Jennifer G. Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 902–911.

[44] Mingzhe Chen, Deniz Gündüz, Kaibin Huang, Walid Saad, Mehdi Bennis, Aneta Vulgarakis Feljan, and H. Vincent Poor. 2021. Distributed learning in wireless networks: Recent progress and future challenges. *IEEE Journal on Selected Areas in Communications* 39, 12 (2021), 3579–3605.

[45] Yudong Chen, Lili Su, and Jiaming Xu. 2017. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 2 (2017), 1–25.

[46] Hui Han Chin, Aleksander Madry, Gary L. Miller, and Richard Peng. 2013. Runtime guarantees for regression problems. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science (ITCS'13)*. ACM, New York, NY, 269–282. https://doi.org/10.1145/2422436.2422469

[47] Michelle S. Chong, Masashi Wakaiki, and Joao P. Hespanha. 2015. Observability of linear systems under adversarial attacks. In *Proceedings of the American Control Conference*. IEEE, Los Alamitos, CA, 2439–2444.

[48] Jack Clark. 2015. Why 2015 was a breakthrough year in artificial intelligence. *Bloomberg Technology (Online Journal)* 18, 1 (2015).

[49] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. In *Proceedings of the 36th International Conference on Machine Learning*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. 1310–1320. https://proceedings.mlr.press/v97/cohen19c.html

[50] Michael B. Cohen, Yin Tat Lee, Gary Miller, Jakub Pachocki, and Aaron Sidford. 2016. Geometric median in nearly linear time. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing*. 9–21.

[51] Georgios Damaskinos, El Mahdi El Mhamdi, Rachid Guerraoui, Rhicheek Patra, and Mahsa Taziki. 2018. Asynchronous Byzantine machine learning (the case of SGD). In *Proceedings of the International Conference on Machine Learning (ICML'18)*. 1145–1154.

[52] Deepesh Data, Linqi Song, and Suhas N. Diggavi. 2020. Data encoding for Byzantine-resilient distributed optimization. *IEEE Transactions on Information Theory* 67, 2 (2020), 1117–1140.

[53] Christopher M. De Sa, Ce Zhang, Kunle Olukotun, and Christopher Ré. 2015. Taming the wild: A unified analysis of hogwild-style algorithms. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15)*.

[54] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. 2012. Large scale distributed deep networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS'12)*. 1223–1231.

[55] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. 2014. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS'14)*. 1646–1654.

[56] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. 2019. Robust estimators in high-dimensions without the computational intractability. *SIAM Journal on Computing* 48, 2 (2019), 742–864.

[57] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. 2019. Sever: A robust meta-algorithm for stochastic optimization. In *Proceedings of the International Conference on Machine Learning (ICML'19)*. 1596–1606.

[58] Ilias Diakonikolas, Daniel M. Kane, and Alistair Stewart. 2017. Statistical query lower bounds for robust estimation of high-dimensional Gaussians and Gaussian mixtures. In *Proceedings of the 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS'17)*. IEEE, Los Alamitos, CA, 73–84.

[59] Aymeric Dieuleveut and Kumar Kshitij Patel. 2019. Communication trade-offs for local-SGD with large step size. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, 1–12. https://proceedings.neurips.cc/paper/2019/file/4aadd661908b181d059a117f02fbc9ec-Paper.pdf

[60] Praveen Kumar Donepudi. 2017. AI and machine learning in banking: A systematic literature review. *Asian Journal of Applied Science and Engineering* 6, 3 (2017), 157–162.

[61] Anran Du, Yicheng Shen, Qinzi Zhang, Lewis Tseng, and Moayad Aloqaily. 2021. CRACAU: Byzantine machine learning meets industrial edge computing in industry 5.0. *IEEE Transactions on Industrial Informatics* 18, 8 (2021), 5435–5445.

[62] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, 7 (2011), 2121–2159.

[63] John C. Duchi, Alekh Agarwal, and Martin J. Wainwright. 2011. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic Control* 57, 3 (2011), 592–606.
[64] El Mahdi El Mhamdi. 2020. *Robust Distributed Learning*. Technical Report. EPFL.
[65] El Mahdi El Mhamdi, Sadegh Farhadkhani, Rachid Guerraoui, Arsany Guirguis, Lê Nguyên Hoang, and Sébastien Rouault. 2021. Collaborative Learning in the Jungle (decentralized, Byzantine, heterogeneous, asynchronous and nonconvex learning). In *Proceedings of the 35th Conference on Neural Information Processing Systems (NIPS'21).*
[66] El Mahdi El Mhamdi, Rachid Guerraoui, Arsany Guirguis, Lê Nguyên Hoang, and Sébastien Rouault. 2020. Genuinely distributed Byzantine machine learning. In *Proceedings of the 39th Symposium on Principles of Distributed Computing.* 355–364.
[67] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. 2018. The hidden vulnerability of distributed learning in Byzantium. *arXiv:stat.ML/1802.07927* (2018).
[68] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. 2021. Distributed momentum for Byzantine-resilient stochastic gradient descent. In *Proceedings of the 9th International Conference on Learning Representations (ICLR'21).*
[69] Cheng Fang, Zhixiong Yang, and Waheed U. Bajwa. 2022. BRIDGE: Byzantine-resilient decentralized gradient descent. *IEEE Transactions on Signal and Information Processing over Networks* 8 (2022), 610–626.
[70] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local model poisoning attacks to Byzantine-robust federated learning. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security'20).* 1605–1622.
[71] Farzam Fanitabasi. 2018. A review of adversarial behaviour in distributed multi-agent optimisation. In *Proceedings of the 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion'18).* IEEE, Los Alamitos, CA, 53–58.
[72] Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Lê-Nguyên Hoang, Rafael Pinot, and John Stephan. 2023. Robust collaborative learning with linear gradient overhead. In *Proceedings of the 40th International Conference on Machine Learning*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.), Vol. 202. PMLR, 9761–9813. https://proceedings.mlr.press/v202/farhadkhani23a.html
[73] Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Rafael Pinot, and John Stephan. 2022. Byzantine machine learning made easy by resilient averaging of momentums. In *Proceedings of the International Conference on Machine Learning (ICML'22).* 6246–6283.
[74] Sadegh Farhadkhani, Rachid Guerraoui, Lê Nguyên Hoang, and Oscar Villemaud. 2022. An equivalence between data poisoning and Byzantine gradient attacks. In *Proceedings of the International Conference on Machine Learning (ICML'22).* 6284–6323.
[75] Jiashi Feng, Huan Xu, and Shie Mannor. 2015. Distributed robust learning. *arXiv:stat.ML/1409.5937* (2015).
[76] Jiashi Feng, Huan Xu, and Shie Mannor. 2017. Outlier robust online learning. *CoRR abs/1701.00251* (2017).
[77] Carlos Poncinelli Filho, Elias Marques Jr., Victor Chang, Leonardo Dos Santos, Flavia Bernardini, Paulo F. Pires, Luiz Ochi, and Flavia C. Delicato. 2022. A systematic literature review on distributed machine learning in edge computing. *Sensors* 22, 7 (2022), 2665.
[78] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. 1985. Impossibility of distributed consensus with one faulty process. *Journal of the ACM* 32, 2 (1985), 374–382.
[79] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. 2020. The limitations of federated learning in sybil settings. In *Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions, and Defenses (RAID'20).* 301–316.
[80] Saeed Ghadimi and Guanghui Lan. 2013. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization* 23, 4 (2013), 2341–2368.
[81] Avishek Ghosh, Raj Kumar Maity, Swanand Kadhe, Arya Mazumdar, and Kannan Ramchandran. 2021. Communication-efficient and Byzantine-robust distributed learning with error feedback. *IEEE Journal on Selected Areas in Information Theory* 2, 3 (2021), 942–953.
[82] Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research* 57 (2016), 345–420.
[83] Eduard Gorbunov, Alexander Borzunov, Michael Diskin, and Max Ryabinin. 2022. Secure distributed training at scale. In *Proceedings of the International Conference on Machine Learning (ICML'22).* 7679–7739.
[84] Eduard Gorbunov, Konstantin P. Burlachenko, Zhize Li, and Peter Richtárik. 2021. MARINA: Faster non-convex distributed learning with compression. In *Proceedings of the International Conference on Machine Learning (ICML'21).* 3788–3798.
[85] Eduard Gorbunov, Samuel Horváth, Peter Richtárik, and Gauthier Gidel. 2023. Variance reduction is an antidote to Byzantines: Better rates, weaker assumptions and communication compression as a cherry on the top. In *Proceedings of the 11th International Conference on Learning Representations.* https://openreview.net/forum?id=pfuqQQCB34
[86] Arnaud Grivet Sébert, Rafael Pinot, Martin Zuber, Cédric Gouy-Pailler, and Renaud Sirdey. 2021. SPEED: Secure, PrivatE, and efficient deep learning. *Machine Learning* 110 (2021), 675–694.

[87] Renjie Gu, Shuo Yang, and Fan Wu. 2019. Distributed machine learning on mobile devices: A survey. *arXiv e-prints arXiv:abs/1909.08329* (2019).

[88] Rachid Guerraoui, Arsany Guirguis, Jérémy Plassmann, Anton Ragot, and Sébastien Rouault. 2021. Garfield: System support for Byzantine machine learning (regular paper). In *Proceedings of the 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'21)*. IEEE, Los Alamitos, CA, 39–51.

[89] Rachid Guerraoui, Nirupam Gupta, Rafaël Pinot, Sébastien Rouault, and John Stephan. 2021. Differential privacy and Byzantine resilience in SGD: Do they add up? In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*. 391–401.

[90] Arsany Hany Abdelmessih Guirguis. 2022. *System Support for Robust Distributed Learning*. Technical Report. EPFL.

[91] Nirupam Gupta, Thinh T. Doan, and Nitin H. Vaidya. 2021. Byzantine fault-tolerance in decentralized optimization under 2f-redundancy. In *Proceedings of the 2021 American Control Conference (ACC'21)*. IEEE, Los Alamitos, CA, 3632–3637.

[92] Nirupam Gupta, Shuo Liu, and Nitin Vaidya. 2021. Byzantine fault-tolerant distributed machine learning with norm-based comparative gradient elimination. In *Proceedings of the 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W'21)*. IEEE, Los Alamitos, CA, 175–181.

[93] Nirupam Gupta and Nitin H. Vaidya. 2019. Byzantine fault-tolerant parallelized stochastic gradient descent for linear regression. In *Proceedings of the 2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton'19)*. IEEE, Los Alamitos, CA, 415–420.

[94] Nirupam Gupta and Nitin H. Vaidya. 2019. Randomized reactive redundancy for Byzantine fault-tolerance in parallelized learning. *arXiv preprint arXiv:1912.09528* (2019).

[95] Nirupam Gupta and Nitin H. Vaidya. 2020. Fault-tolerance in distributed optimization: The case of redundancy. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*. 365–374.

[96] Nirupam Gupta and Nitin H. Vaidya. 2020. Resilience in collaborative optimization: Redundant and independent cost functions. *arXiv preprint arXiv:2003.09675* (2020).

[97] Nirupam Gupta and Nitin H. Vaidya. 2021. Byzantine fault-tolerance in peer-to-peer distributed gradient-descent. *arXiv preprint arXiv:2101.12316* (2021).

[98] J. B. S. Haldane. 1948. Note on the median of a multivariate distribution. *Biometrika* 35, 3-4 (1948), 414–417.

[99] Anand Handa, Ashu Sharma, and Sandeep K. Shukla. 2019. Machine learning in cybersecurity: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9, 4 (2019), e1306.

[100] Lie He, Sai Praneeth Karimireddy, and Martin Jaggi. 2022. Byzantine-robust decentralized learning via self-centered clipping. *CoRR abs/2202.01545* (2022). https://arxiv.org/abs/2202.01545

[101] Marijn Hoijtink and Anneroos Planqué-van Hardeveld. 2022. Machine learning and the platformization of the military: A study of Google's machine learning platform TensorFlow. *International Political Sociology* 16, 2 (2022), olab036.

[102] Rui Hu, Yuanxiong Guo, Hongning Li, Qingqi Pei, and Yanmin Gong. 2020. Personalized federated learning with differential privacy. *IEEE Internet of Things Journal* 7, 10 (2020), 9530–9539.

[103] Shuyan Hu, Xiaojing Chen, Wei Ni, Ekram Hossain, and Xin Wang. 2021. Distributed machine learning for wireless communication networks: Techniques, architectures, and applications. *IEEE Communications Surveys & Tutorials* 23, 3 (2021), 1458–1493. https://doi.org/10.1109/COMST.2021.3086014

[104] Peter J. Huber. 2011. Robust statistics. In *International Encyclopedia of Statistical Science*. Springer, 1248–1251.

[105] Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I. Jordan. 2014. Communication-efficient distributed dual coordinate ascent. *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS'14)*. 3068–3076.

[106] Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. 1986. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science* 43 (1986), 169–188.

[107] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2021. *Advances and Open Problems in Federated Learning*. Now Foundations and Trends. https://doi.org/10.1561/2200000083

[108] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. 2021. Byzantine-robust learning on heterogeneous datasets via bucketing. In *Proceedings of the International Conference on Learning Representations*.

[109] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. 2021. Learning from history for Byzantine robust optimization. In *Proceedings of the International Conference on Machine Learning*, Vol. 139.

[110] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *Proceedings of the International Conference on Machine Learning (ICML'20)*. 5132–5143.

[111] Paramjit Kaur, Kewal Krishan, Suresh K. Sharma, and Tanuj Kanchan. 2020. Facial-recognition algorithms: A literature review. *Medicine, Science and the Law* 60, 2 (2020), 131–139.

[112] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15): Conference Track*. http://arxiv.org/abs/1412.6980

[113] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian U. Stich. 2020. A unified theory of decentralized SGD with changing topology and local updates. In *Proceedings of the 37th International Conference on Machine Learning*, Hal Dauma III and Aarti Singh (Eds.), Vol. 119. PMLR, 5381–5393. http://proceedings.mlr.press/v119/koloskova20a.html

[114] Kananart Kuwaranancharoen, Lei Xin, and Shreyas Sundaram. 2020. Byzantine-resilient distributed optimization of multi-dimensional functions. In *Proceedings of the 2020 American Control Conference (ACC'20)*. IEEE, Los Alamitos, CA, 4399–4404.

[115] Kevin A. Lai, Anup B. Rao, and Santosh Vempala. 2016. Agnostic estimation of mean and covariance. In *Proceedings of the 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS'16)*. IEEE, Los Alamitos, CA, 665–674.

[116] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems* 4, 3 (July 1982), 382–401. https://doi.org/10.1145/357172.357176

[117] Heath J. LeBlanc, Haotian Zhang, Xenofon Koutsoukos, and Shreyas Sundaram. 2013. Resilient asymptotic consensus in robust networks. *IEEE Journal on Selected Areas in Communications* 31, 4 (2013), 766–781.

[118] Martin Leo, Suneel Sharma, and Koilakuntla Maddulety. 2019. Machine learning in banking risk management: A literature review. *Risks* 7, 1 (2019), 29.

[119] Annick M. Leroy and Peter J. Rousseeuw. 1987. *Robust Regression and Outlier Detection*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons.

[120] 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 (OSDI'14)*. 583–598.

[121] Mu Li, David G. Andersen, Alexander J. Smola, and Kai Yu. 2014. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (Eds.), Vol. 27. Curran Associates, 1–16. https://proceedings.neurips.cc/paper/2014/file/1ff1de774005f8da13f42943881c655f-Paper.pdf

[122] Min Li, Di Xiao, Jia Liang, and Hui Huang. 2022. Communication-efficient and Byzantine-robust differentially private federated learning. *IEEE Communications Letters* 26, 8 (2022), 1725–1729.

[123] Mu Li, Li Zhou, Zichao Yang, Aaron Li, Fei Xia, David G. Andersen, and Alexander Smola. 2013. Parameter server for distributed machine learning. In *Proceedings of the Big Learning NIPS Workshop*, Vol. 6.

[124] Suyi Li, Yong Cheng, Yang Liu, Wei Wang, and Tianjian Chen. 2019. Abnormal client behavior detection in federated learning. *arXiv preprint arXiv:1910.09933* (2019).

[125] Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. 2015. Asynchronous parallel stochastic gradient for nonconvex optimization. *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15)*. 2737–2745.

[126] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. 2017. Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. 5336–5346.

[127] Ji Liu, Jizhou Huang, Yang Zhou, Xuhong Li, Shilei Ji, Haoyi Xiong, and Dejing Dou. 2022. From distributed machine learning to federated learning: A survey. *Knowledge and Information Systems* 64 (2022), 1–33.

[128] Shuo Liu. 2021. A survey on fault-tolerance in distributed optimization and machine learning. *arXiv preprint arXiv:2106.08545* (2021).

[129] Shuo Liu, Nirupam Gupta, and Nitin H. Vaidya. 2021. Approximate Byzantine fault-tolerance in distributed optimization. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC'21)*. ACM, New York, NY, 379–389. https://doi.org/10.1145/3465084.3467902

[130] Zhuo Lv, Hongbo Cao, Feng Zhang, Yuange Ren, Bin Wang, Cen Chen, Nuannuan Li, Hao Chang, and Wei Wang. 2022. AWFC: Preventing label flipping attacks towards federated learning for intelligent IoT. *Computer Journal* 65, 11 (2022), 2849–2859.

[131] Nancy A. Lynch. 1996. *Distributed Algorithms*. Elsevier.

[132] Xu Ma, Xiaoqian Sun, Yuduo Wu, Zheli Liu, Xiaofeng Chen, and Changyu Dong. 2022. Differentially private Byzantine-robust federated learning. *IEEE Transactions on Parallel and Distributed Systems* 33, 12 (2022), 3690–3701.
[133] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *Proceedings of the International Conference on Learning Representations.*
[134] Saeed Mahloujifar, Mohammad Mahmoody, and Ameer Mohammed. 2019. Data poisoning attacks in multi-party learning. In *Proceedings of the International Conference on Machine Learning (ICML'19).* 4274–4283.
[135] Yanwen Mao, Aritra Mitra, Shreyas Sundaram, and Paulo Tabuada. 2019. When is the secure state-reconstruction problem hard? In *Proceedings of the 2019 IEEE 58th Conference on Decision and Control (CDC'19).* IEEE, Los Alamitos, CA, 5368–5373.
[136] Laurent Meunier, Meyer Scetbon, Rafael B. Pinot, Jamal Atif, and Yann Chevaleyre. 2021. Mixed Nash equilibria in the adversarial examples game. In *Proceedings of the International Conference on Machine Learning (ICML'21).* 7677–7687.
[137] Shaunak Mishra, Yasser Shoukry, Nikhil Karamchandani, Suhas N. Diggavi, and Paulo Tabuada. 2016. Secure state estimation against sensor attacks in the presence of noise. *IEEE Transactions on Control of Network Systems* 4, 1 (2016), 49–59.
[138] Aritra Mitra, Rayana Jaafar, George J. Pappas, and Hamed Hassani. 2021. Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS'21).*
[139] Aritra Mitra and Shreyas Sundaram. 2019. Byzantine-resilient distributed observers for LTI systems. *Automatica* 108 (2019), 108487.
[140] Luis Muñoz-González, Kenneth T. Co, and Emil C. Lupu. 2019. Byzantine-robust federated machine learning through adaptive model averaging. *arXiv preprint arXiv:1909.05125* (2019).
[141] Dmitry Namiot, Eugene Ilyushin, and Ivan Chizhov. 2021. Military applications of machine learning. *International Journal of Open Information Technologies* 10, 1 (2021), 69–76.
[142] Meenal V. Narkhede, Prashant P. Bartakke, and Mukul S. Sutaone. 2022. A review on weight initialization strategies for neural networks. *Artificial Intelligence Review* 55, 1 (2022), 291–322. https://doi.org/10.1007/s10462-021-10033-z
[143] Angelia Nedic and Asuman Ozdaglar. 2009. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control* 54, 1 (2009), 48–61.
[144] A. S. Nemirovsky, D. B. Din, and D. B. Yudin. 1983. *Problem Complexity and Method Efficiency in Optimization.* Wiley. https://books.google.ch/books?id=6ULvAAAAMAAJ
[145] Maxence Noble, Aurélien Bellet, and Aymeric Dieuleveut. 2022. Differentially private federated learning on heterogeneous data. In *Proceedings of the International Conference on Artificial Intelligence and Statistics.* 10110–10145.
[146] Miroslav Pajic, Insup Lee, and George J. Pappas. 2017. Attack-resilient state estimation for noisy dynamical systems. *IEEE Transactions on Control of Network Systems* 4, 1 (2017), 82–92.
[147] Miroslav Pajic, James Weimer, Nicola Bezzo, Paulo Tabuada, Oleg Sokolsky, Insup Lee, and George J. Pappas. 2014. Robustness of attack-resilient state estimators. In *Proceedings of the ACM/IEEE 5th International Conference on Cyber-Physical Systems (with CPS Week'14) (ICCPS'14).* IEEE, Los Alamitos, CA, 163–174.
[148] Pablo A. Parrilo and Bernd Sturmfels. 2003. Minimizing polynomial functions. *Algorithmic and Quantitative Real Algebraic Geometry, DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 60 (2003), 83–99.
[149] Diego Peteiro-Barral and Bertha Guijarro-Berdiñas. 2013. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence* 2, 1 (2013), 1–11.
[150] Yulu Pi. 2021. Machine learning in governments: Benefits, challenges and future directions. *eJournal of eDemocracy and Open Government* 13, 1 (2021), 203–219.
[151] Krishna Pillutla, Sham M. Kakade, and Zaid Harchaoui. 2022. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing* 70 (2022), 1142–1154. https://doi.org/10.1109/TSP.2022.3153135
[152] Rafael Pinot, Raphael Ettedgui, Geovani Rizk, Yann Chevaleyre, and Jamal Atif. 2020. Randomization matters. How to defend against strong adversarial attacks. In *Proceedings of the International Conference on Machine Learning (ICML'20).*
[153] Rafael Pinot, Laurent Meunier, Alexandre Araujo, Hisashi Kashima, Florian Yger, Cédric Gouy-Pailler, and Jamal Atif. 2019. Theoretical evidence for adversarial robustness through randomization. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS'19).* 11860–11870.
[154] Rafael Pinot, Laurent Meunier, Florian Yger, Cédric Gouy-Pailler, Yann Chevaleyre, and Jamal Atif. 2022. On the robustness of randomized classifiers to adversarial examples. *Machine Learning* 111, 9 (2022), 3425–3457.
[155] Boris Polyak. 1964. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics* 4 (1964), 1–17. https://doi.org/10.1016/0041-5553(64)90137-5
[156] Adarsh Prasad, Arun Sai Suggala, Sivaraman Balakrishnan, and Pradeep Ravikumar. 2020. Robust estimation via robust gradient estimation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 82, 3 (2020), 601–627.

[157] S. Preetha, I. M. Afrid, P. Karthik Hebbar, and S. K. Nishchay. 2020. Machine learning for handwriting recognition. *International Journal of Computer* 38, 1 (2020), 93–101.

[158] Adnan Qayyum, Junaid Qadir, Muhammad Bilal, and Ala Al-Fuqaha. 2020. Secure and robust machine learning for healthcare: A survey. *IEEE Reviews in Biomedical Engineering* 14 (2020), 156–180.

[159] Shashank Rajput, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. 2019. DETOX: A redundancy-based framework for faster and more robust gradient aggregation. In *Proceedings of the International Conference on Machine Learning (ICML'19)*.

[160] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS'11)*. 693–701.

[161] Jayanth Regatti, Hao Chen, and Abhishek Gupta. 2020. ByGARS: Byzantine SGD with arbitrary number of attackers. *arXiv preprint arXiv:2006.13421* (2020).

[162] Xiaoqiang Ren, Yilin Mo, Jie Chen, and Karl Henrik Johansson. 2020. Secure state estimation with Byzantine sensors: A probabilistic approach. *IEEE Transactions on Automatic Control* 65, 9 (2020), 3742–3757.

[163] Gernot Rieder and Judith Simon. 2016. Datatrust: Or, the political quest for numerical evidence and the epistemologies of Big Data. *Big Data & Society* 3, 1 (2016), 2053951716649398.

[164] Thomas J. Rothenberg, Franklin M. Fisher, and Christian Bernhard Tilanus. 1964. A note on estimation from a Cauchy sample. *Journal of the American Statistical Association* 59, 306 (1964), 460–463.

[165] Sébastien Louis Alexandre Rouault. 2022. *Practical Byzantine-Resilient Stochastic Gradient Descent.* Technical Report. EPFL.

[166] Peter J. Rousseeuw. 1985. Multivariate estimation with high breakdown point. *Mathematical Statistics and Applications* 8, 37 (1985), 283–297.

[167] Stuart J. Russell. 2010. *Artificial Intelligence: A Modern Approach*. Pearson Education.

[168] David Saldana, Amanda Prorok, Shreyas Sundaram, Mario F. M. Campos, and Vijay Kumar. 2017. Resilient consensus for time-varying networks of dynamic agents. In *Proceedings of the 2017 American Control Conference (ACC'17)*. IEEE, Los Alamitos, CA, 252–258.

[169] K. Shailaja, B. Seetharamulu, and M. A. Jabbar. 2018. Machine learning in healthcare: A review. In *Proceedings of the 2018 2nd International Conference on Electronics, Communication, and Aerospace Technology (ICECA'18)*. IEEE, Los Alamitos, CA, 910–914.

[170] Virat Shejwalkar and Amir Houmansadr. 2021. Manipulating the Byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *Proceedings of the 2021 Network and Distributed System Security Symposium (NDSS'21)*.

[171] Junyu Shi, Wei Wan, Shengshan Hu, Jianrong Lu, and Leo Yu Zhang. 2022. Challenges and approaches for mitigating Byzantine attacks in federated learning. In *Proceedings of the 2022 IEEE International Conference on Trust, Security, and Privacy in Computing and Communications (TrustCom'22)*. IEEE, Los Alamitos, CA, 139–146.

[172] Shaohuai Shi, Zhenheng Tang, Xiaowen Chu, Chengjian Liu, Wei Wang, and Bo Li. 2021. A quantitative survey of communication optimizations in distributed deep learning. *IEEE Network* 35, 3 (2021), 230–237. https://doi.org/10.1109/MNET.011.2000530

[173] Yuanming Shi, Kai Yang, Tao Jiang, Jun Zhang, and Khaled B. Letaief. 2020. Communication-efficient edge AI: Algorithms and systems. *IEEE Communications Surveys & Tutorials* 22, 4 (2020), 2167–2191. https://doi.org/10.1109/COMST.2020.3007787

[174] Yasser Shoukry, Pierluigi Nuzzo, Alberto Puggelli, Alberto L. Sangiovanni-Vincentelli, Sanjit A. Seshia, Mani Srivastava, and Paulo Tabuada. 2015. Imhotep-SMT: A satisfiability modulo theory solver for secure state estimation. In *Proceedings of the International Workshop on Satisfiability Modulo Theories*. 3–13.

[175] Yasser Shoukry, Pierluigi Nuzzo, Alberto Puggelli, Alberto L. Sangiovanni-Vincentelli, Sanjit A. Seshia, and Paulo Tabuada. 2017. Secure state estimation for cyber-physical systems under sensor attacks: A satisfiability modulo theory approach. *IEEE Transactions on Automatic Control* 62, 10 (2017), 4917–4932.

[176] Christopher G. Small. 1990. A survey of multidimensional medians. *International Statistical Review/Revue Internationale de Statistique* 58, 3 (1990), 263–277.

[177] Jacob Steinhardt, Moses Charikar, and Gregory Valiant. 2018. Resilience: A criterion for learning in the presence of arbitrary outliers. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference (ITCS'18)*.

[178] Lili Su and Shahin Shahrampour. 2019. Finite-time guarantees for Byzantine-resilient distributed state estimation with noisy measurements. *IEEE Transactions on Automatic Control* 65, 9 (2019), 3758–3771.

[179] Lili Su and Nitin Vaidya. 2016. Multi-agent optimization in the presence of Byzantine adversaries: Fundamental limits. In *Proceedings of the American Control Conference (ACC'16)*. IEEE, Los Alamitos, CA, 7183–7188.

[180] Lili Su and Nitin H. Vaidya. 2016. Fault-tolerant multi-agent optimization: Optimal iterative distributed algorithms. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*. 425–434.

[181] Lili Su and Nitin H. Vaidya. 2016. Non-Bayesian learning in the presence of Byzantine agents. In *Distributed Computing*. Springer, Berlin, Germany, 414–427.

[182] Lili Su and Nitin H. Vaidya. 2016. Robust multi-agent optimization: Coping with Byzantine agents with input redundancy. In *Proceedings of the International Symposium on Stabilization, Safety, and Security of Distributed Systems*. 368–382.

[183] Lili Su and Nitin H. Vaidya. 2021. Byzantine-resilient multiagent optimization. *IEEE Transactions on Automatic Control* 66, 5 (2021), 2227–2233.

[184] Shreyas Sundaram and Bahman Gharesifard. 2015. Consensus-based distributed optimization with malicious nodes. In *Proceedings of the 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton'15)*. IEEE, Los Alamitos, CA, 244–249.

[185] Shreyas Sundaram and Bahman Gharesifard. 2018. Distributed optimization under adversarial nodes. *IEEE Transactions on Automatic Control* 64, 3 (2018), 1063–1076.

[186] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations*.

[187] Shanjiang Tang, Bingsheng He, Ce Yu, Yusen Li, and Kun Li. 2022. A survey on spark ecosystem: Big data processing infrastructure, machine learning, and applications. *IEEE Transactions on Knowledge and Data Engineering* 34, 1 (2022), 71–91. https://doi.org/10.1109/TKDE.2020.2975652

[188] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. 2020. On adaptive attacks to adversarial example defenses. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, 1633–1645. https://proceedings.neurips.cc/paper/2020/file/11f38f8ecd71867b42433548d1078e38-Paper.pdf

[189] Thi Ngoc Trang Tran, Alexander Felfernig, Christoph Trattner, and Andreas Holzinger. 2021. Recommender systems in the healthcare domain: State-of-the-art and research issues. *Journal of Intelligent Information Systems* 57, 1 (2021), 171–201.

[190] Ayushi Trivedi, Navya Pant, Pinal Shah, Simran Sonik, and Supriya Agrawal. 2018. Speech to text and text to speech recognition systems—A review. *IOSR Journal of Computer Engineering* 20, 2 (2018), 36–43.

[191] Lewis Tseng and Nitin Vaidya. 2013. Iterative approximate Byzantine consensus under a generalized fault model. In *Proceedings of the International Conference on Distributed Computing and Networking*. 72–86.

[192] Lewis Tseng and Nitin H. Vaidya. 2015. Fault-tolerant consensus in directed graphs. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*. 451–460.

[193] Nitin Vaidya. 2012. Matrix representation of iterative approximate Byzantine consensus in directed graphs. *arXiv preprint arXiv:1203.1888* (2012).

[194] Nitin H. Vaidya. 2014. Iterative Byzantine vector consensus in incomplete graphs. In *Proceedings of the International Conference on Distributed Computing and Networking*. 14–28.

[195] Nitin H. Vaidya, Lewis Tseng, and Guanfeng Liang. 2012. Iterative approximate Byzantine consensus in arbitrary directed graphs. In *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing*. 365–374.

[196] Colin van Noordt and Gianluca Misuraca. 2022. Exploratory insights on artificial intelligence for government in Europe. *Social Science Computer Review* 40, 2 (2022), 426–444.

[197] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer. 2020. A survey on distributed machine learning. *ACM Computing Surveys* 53, 2 (2020), 1–33.

[198] Thijs Vogels, Hadrien Hendrikx, and Martin Jaggi. 2022. Beyond spectral gap: The role of the topology in decentralized learning. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS'22)*. http://papers.nips.cc/paper_files/paper/2022/hash/61162d94822d468ee6e92803340f2040-Abstract-Conference.html

[199] Wei Wan, Jianrong Lu, Shengshan Hu, Leo Yu Zhang, and Xiaobing Pei. 2021. Shielding federated learning: A new attack approach and its defense. In *Proceedings of the 2021 IEEE Wireless Communications and Networking Conference (WCNC'21)*. IEEE, Los Alamitos, CA, 1–7.

[200] Jianyu Wang and Gauri Joshi. 2019. Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD. *Proceedings of Machine Learning and Systems* 1 (2019), 212–229.

[201] Meng Wang, Weijie Fu, Xiangnan He, Shijie Hao, and Xindong Wu. 2022. A survey on large-scale machine learning. *IEEE Transactions on Knowledge and Data Engineering* 34, 6 (2022), 2574–2594. https://doi.org/10.1109/TKDE.2020.3015777

[202] Wei Wang and Jianxun Gang. 2018. Application of convolutional neural network in natural language processing. In *Proceedings of the 2018 International Conference on Information Systems and Computer Aided Education (ICISCAE'18)*. IEEE, Los Alamitos, CA, 64–70.

[203] Zhaoxian Wu, Qing Ling, Tianyi Chen, and Georgios B. Giannakis. 2020. Federated variance-reduced stochastic gradient descent with robustness to Byzantine attacks. *IEEE Transactions on Signal Processing* 68 (2020), 4583–4596.

[204] Qi Xia, Zeyi Tao, Zijiang Hao, and Qun Li. 2019. FABA: An algorithm for fast aggregation against Byzantine attacks in distributed neural networks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19).*

[205] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. 2018. Generalized Byzantine-tolerant SGD. *arXiv:cs.DC/1802.10116* (2018).

[206] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. 2019. Fall of empires: Breaking Byzantine-tolerant SGD by inner product manipulation. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI'19).* 83.

[207] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. 2019. SLSGD: Secure and efficient distributed on-device machine learning. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* 213–228.

[208] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2019. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *Proceedings of the 36th International Conference on Machine Learning*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, 6893–6901.

[209] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2020. Zeno++: Robust fully asynchronous SGD. In *Proceedings of the International Conference on Machine Learning (ICML'20).* 10495–10503.

[210] Tao Yang, Xinlei Yi, Junfeng Wu, Ye Yuan, Di Wu, Ziyang Meng, Yiguang Hong, Hong Wang, Zongli Lin, and Karl H. Johansson. 2019. A survey of distributed optimization. *Annual Reviews in Control* 47 (2019), 278–305.

[211] Zhixiong Yang and Waheed U. Bajwa. 2019. ByRDiE: Byzantine-resilient distributed coordinate descent for decentralized learning. *IEEE Transactions on Signal and Information Processing over Networks* 5, 4 (2019), 611–627.

[212] Zhixiong Yang, Arpita Gang, and Waheed U. Bajwa. 2020. Adversary-resilient distributed and decentralized statistical inference and machine learning: An overview of recent advances under the Byzantine threat model. *IEEE Signal Processing Magazine* 37, 3 (2020), 146–159.

[213] Xin Yao, Tianchi Huang, Rui-Xiao Zhang, Ruiyu Li, and Lifeng Sun. 2019. Federated learning with unbiased gradient aggregation and controllable meta updating. *arXiv preprint arXiv:1910.08234* (2019).

[214] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proceedings of the International Conference on Machine Learning (ICML'18).* 5650–5659.

[215] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2019. Defending against saddle point attack in Byzantine-robust distributed learning. In *Proceedings of the International Conference on Machine Learning (ICML'19).* 7074–7084.

[216] Hao Yu and Rong Jin. 2019. On the computation and communication complexity of parallel SGD with dynamic batch sizes for stochastic non-convex optimization. In *Proceedings of the 36th International Conference on Machine Learning*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, 7174–7183. https://proceedings.mlr.press/v97/yu19c.html

[217] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. 2020. BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC'20).* 493–506.

[218] Haotian Zhang and Shreyas Sundaram. 2012. Robustness of complex networks with implications for consensus and contagion. In *Proceedings of the 2012 IEEE 51st IEEE Conference on Decision and Control (CDC'12).* IEEE, Los Alamitos, CA, 3426–3432.

[219] Haotian Zhang and Shreyas Sundaram. 2012. Robustness of information diffusion algorithms to locally bounded adversaries. In *Proceedings of the 2012 American Control Conference (ACC'12).* IEEE, Los Alamitos, CA, 5855–5861.

[220] Qinzi Zhang and Lewis Tseng. 2021. Echo-CGC: A communication-efficient Byzantine-tolerant distributed machine learning algorithm in single-hop radio network. In *Proceedings of the 24th International Conference on Principles of Distributed Systems (OPODIS'20).*

[221] Zhaoning Zhang, Lujia Yin, Yuxing Peng, and Dongsheng Li. 2018. A quick survey on large scale distributed deep learning systems. In *Proceedings of the 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS'18).* IEEE, Los Alamitos, CA, 1052–1056.

[222] Shifa Zhong, Kai Zhang, Majid Bagheri, Joel G. Burken, April Gu, Baikun Li, Xingmao Ma, Babetta L. Marrone, Zhiyong Jason Ren, Joshua Schrier, Wei Shi, Haoyue Tan, Tianbao Wang, Xu Wang, Bryan M. Wong, Xusheng Xiao, Xiong Yu, Jun-Jie Zhu, and Huichun Zhang. 2021. Machine learning: New ideas and tools in environmental science and engineering. *Environmental Science & Technology* 55, 19 (2021), 12741–12754.

[223] Banghua Zhu, Lun Wang, Qi Pang, Shuai Wang, Jiantao Jiao, Dawn Song, and Michael I. Jordan. 2023. Byzantine-robust federated learning with optimal statistical rates. In *Proceedings of the International Conference on Artificial Intelligence and Statistics.* 3151–3178.

[224] Heng Zhu and Qing Ling. 2022. Bridging differential privacy and Byzantine-robustness via model aggregation. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI'22)*. 2427–2433. https://doi.org/10.24963/ijcai.2022/337

[225] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex Smola. 2010. Parallelized stochastic gradient descent. *Proceedings of the International Conference on Neural Information Processing Systems (NIPS'10)*.

Received 20 October 2022; revised 30 May 2023; accepted 3 August 2023