Fall of Empires: Breaking Byzantine-tolerant SGD by Inner Product Manipulation

Cong Xie
cx2@illinois.edu

Oluwasanmi Koyejo
sanmi@illinois.edu

Indranil Gupta
indy@illinois.edu

Computer Science Dept., University of Illinois at Urbana-Champaign

Abstract

Recently, new defense techniques have been developed to tolerate Byzantine failures for distributed machine learning. The Byzantine model captures workers that behave arbitrarily, including malicious and compromised workers. In this paper, we break two prevailing Byzantine-tolerant techniques. Specifically we show that two robust aggregation methods for synchronous SGD–namely, coordinate-wise median and Krum–can be broken using new attack strategies based on inner product manipulation. We prove our results theoretically, as well as show empirical validation.

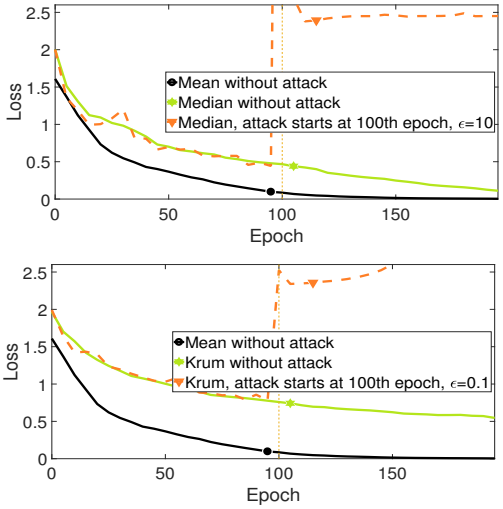


Figure 1: Illustration of failed Byzantine-tolerant SGD. We execute distributed synchronous SGD on CIFAR-10 image classification, with 25 workers. Beginning from the 100th epoch, we attack the system by replacing some workers with Byzantine workers. During the attack, 12 workers are Byzantine for coordinate-wise median, and 11 workers are Byzantine for Krum. The Byzantine workers push $(-\epsilon g)$ to the server, where g is the true gradient.

1 INTRODUCTION

The security of distributed machine learning has drawn increasing attention in recent years. Among the threat models, Byzantine failures (Lamport et al., 1982) are perhaps the most well-studied. In the Byzantine threat model, workers can behave arbitrarily and maliciously. In addition, Byzantine workers are omniscient and can conspire. Most of the existing Byzantine-tolerant machine-learning algorithms (Blanchard et al., 2017; Chen et al., 2017; Yin et al., 2018; Feng et al., 2014; Su & Vaidya, 2016a;b; Alistarh et al., 2018) focus on the protection of distributed Stochastic Gradient Descent (SGD).

In this paper, we consider Byzantine-tolerant SGD in a server-worker architecture (also known as the parameter server architecture (Li et al., 2014a;b)), depicted in Figure 3. The system is composed of server nodes and worker nodes. In each epoch, the workers pull the latest model from the servers, estimate the gradients using the locally sampled training data, and then push the gradient estimators to the servers. The servers aggregate the gradient estimators, and update the model.

We consider Byzantine failures at a subset of the worker nodes. Byzantine workers send arbitrary values instead of the gradient estimators to the server. Such Byzantine gradients are potentially adversarial, and this can result in convergence to sub-optimum models, or even lead to divergence. To make things worse, the Byzantine workers can spy on the information at any server or at any honest worker (omniscience). Byzantine gradients can thus be tailored to have similar variance and magnitude as the correct gradients, which makes them hard to distinguish. Additionally, in different iterations, different subsets of workers can behave in a Byzantine manner, evading detection. Existing literature assumes that less than half of the workers are Byzantine in any iteration.

Compared to traditional Byzantine tolerance in distributed systems (Lynch, 1996; Avizienis et al., 2004; Tanenbaum & Van Steen, 2007; Fischer et al., 1982), Byzantine tolerance in distributed machine learning has unique properties and challenges. Traditional Byzantine tolerance attempts to reach consensus on correct values. However, machine learning algorithms do not need to reach consensus. Further, even non-Byzantine-tolerant machine learning algorithms can naturally tolerate some noise in the input and during execution (Xing et al., 2016). Thus for distributed SGD, existing techniques for Byzantine-tolerant execution guarantee an upper-bound on the distance between the aggregated approximate gradient (under Byzantine workers) and the true gradient (Blanchard et al., 2017; Yin et al., 2018).

A deeper introspection reveals, however, that what really matters for gradient descent algorithms is the direction of the descent. As shown in Figure 2, to ensure the gradient descent algorithm makes progress, we need to guarantee that the direction of the aggregated vector agrees with the true gradient, i.e., the inner product between the aggregated vector and the true gradient must be non-negative. This can be violated by an attack that makes the inner product negative. We call this class of new attacks "inner product manipulation attacks".

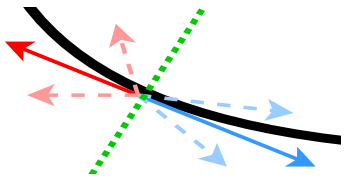


Figure 2: Descent Direction. The blue arrows are the directions which agree with the steepest descent direction (negative gradient). The red arrows are the directions which agree with the steepest ascent direction (gradient).

We observe that the bounded distance between the aggregated value and the true gradient guaranteed by existing techniques is not enough to defend distributed synchronous SGD against inner product manipulation attacks. For example, for the coordinate-wise median, if we put all the Byzantine values on the opposite side of the true gradient, the inner product between the aggregated vector and the true gradient can be manipulated to be negative.

In this paper, we study how inner product manipulation makes Byzantine-tolerant SGD vulnerable. We conduct case studies on coordinate-wise median (Yin et al., 2018) and Krum (Blanchard et al., 2017). Figure 1 gives a glimpse of how bad the effect of the attack can be. In a nutshell, creating gradients in the opposite direction with large magnitude crashes coordinate-wise median, while creating gradients in the opposite direction with small magnitude crashes Krum. We provide theoretical analysis as well as empirical results to validate these findings.

Based on these results, we argue that there is a need to revise the definition of Byzantine tolerance in distributed SGD. We provide a new definition, and study its satisfaction for two popular robust distributed SGD algorithms, theoretically and empirically. In summary, our contributions are:

- We break two popular Byzantine tolerant SGD algorithms – coordinate-wise median (Yin et al., 2018) and Krum (Blanchard et al., 2017) – using a new class of attacks called inner production manipulation attacks. We theoretically prove that under certain conditions, we can backdoor these two algorithms, even when the assumptions and theorems presented in these papers are valid.
- We show how to design Byzantine gradients to compromise the robust aggregation rules. We conduct experiments to validate further.
- Following our theoretical and empirical analysis, we propose a revised definition of Byzantine tolerance for distributed SGD.

2 RELATED WORK

Robust estimators such as the median are well studied, and can naturally be applied to Byzantine tolerance. Coordinate-wise median is one approach that generalizes the median to high-dimensional vectors. In Yin et al. (2018), statistical error rates for the coordinate-wise median in distributed SGD are studied. Xie et al. (2019a) uses trimmed mean for federated optimization.

Blanchard et al. (2017) propose Krum, which is not based on robust statistics. For each candidate gradient, Krum computes the local sum of squared Euclidean distances to the others, and outputs the one with minimal sum.

In this paper, we focus on coordinate-wise median and Krum. There are other Byzantine-tolerant SGD algorithms. For example, Bulyan (Guerraoui et al., 2018) which is based on Krum, and potentially shares the same flaws. DRACO (Chen et al., 2018) uses coding theory to ensure robustness, and is different from the other Byzantine-tolerant SGD algorithms.

Xie et al. (2019b) propose Zeno, which filters out potentially malicious gradients by checking the descent of the loss function. Such a mechanism potentially tolerates the attack techniques introduced in this paper.

Recently, an increasing number of papers have proposed attack mechanisms to break the defenses of machine learn-

ing in settings different from ours. For example, Athalye et al. (2018) propose attack techniques using adversarial training data. Bhagoji et al. (2018); Bagdasaryan et al. (2018) break the defense of federated learning (McMahan et al., 2016). In this paper, we focus on attacking distributed synchronous SGD using adversarial gradients sent by Byzantine workers.

3 PRELIMINARIES

In this paper, we focus on distributed synchronous Stochastic Gradient Descent (SGD) with a Parameter Server (PS). In this section, we formally introduce distributed synchronous SGD and the threat model of Byzantine failures.

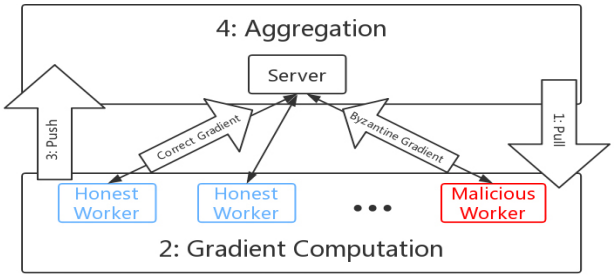


Figure 3: Worker-Server Architecture

3.1 STOCHASTIC GRADIENT DESCENT

We consider the following optimization problem:

$$\min_{x\in\mathbb{R}^d} F(x),$$

where $F(x) = \mathbb{E}_{z\sim\mathcal{D}}[f(x;z)]$ is a differentiable function, z is sampled from some unknown distribution $\mathcal{D}$, d is the number of dimensions. We assume that there exists at least one minimizer of $F(x)$, which is denoted by x^*, where $\nabla F(x^*) = 0$.

This problem is solved in a distributed manner with m workers. In each iteration, each worker will sample n independent and identically distributed (i.i.d.) data points from the distribution $\mathcal{D}$, and compute the gradient of the local empirical loss $F_i(x) = \frac{1}{n}\sum_{j=1}^{n} f(x;z^{i,j}), \forall i \in [m]$, where $z^{i,j}$ is the jth sampled data on the ith worker. The servers will collect and aggregate the gradients sent by the workers, and update the model as follows:

$$x^{t+1} = x^t - \gamma^t \texttt{Aggr}(\{\tilde{v}_i^t : i \in [m]\}),$$

where $\texttt{Aggr}(\cdot)$ is an aggregation rule (e.g., averaging), and $\tilde{v}_i^t$ is the gradient sent by the ith worker, γ^t is the learning rate in the t^{th} iteration. For an honest worker, $\tilde{v}_i^t = \nabla F_i(x^t)$ is an unbiased estimator such that $\mathbb{E}[\nabla F_i(x^t)] = \nabla F(x^t)$. When all the workers are honest, the most common choice of the aggregation rule $\texttt{Aggr}(\cdot)$ is averaging:

$$\texttt{Aggr}(\{\tilde{v}_i^t : i \in [m]\}) = \frac{1}{m}\sum_{i\in[m]} \tilde{v}_i^t.$$

The detailed algorithm of distributed synchronous SGD with aggregation rule $\texttt{Aggr}(\cdot)$ is shown in Algorithm 1.

Algorithm 1 Distributed Synchronous SGD with Robust Aggregation

Server
$x^0 \leftarrow rand()$ {Initialization}
for $t = 0, \ldots, T$ **do**
  Broadcast x^t to all the workers
  Wait until all the gradients $\{\tilde{v}_i^t : i \in [m]\}$ arrive
  Compute $\tilde{\bar{v}}^t = \texttt{Aggr}(\{\tilde{v}_i^t : i \in [m]\})$
  Update the parameter $x^{t+1} \leftarrow x^t - \gamma^t \tilde{\bar{v}}^t$
end for

Worker $i = 1, \ldots, m$
for $t = 0, \ldots, T$ **do**
  Receive x^t from the server
  Draw the samples, compute, and send the gradient $v_i^t = \nabla F_i^t(x^t)$ to the server
end for

3.2 THREAT MODEL

In the Byzantine failure model, the gradients sent by malicious workers can take an arbitrary value:

$$\tilde{v}_i^t = \begin{cases} *, & \text{if } i\text{th worker is Byzantine,} \\ \nabla F_i(x^t), & \text{otherwise,} \end{cases} \quad (1)$$

where "$*$" represents arbitrary values.

Formally, we define the threat model of Byzantine failure as follows.

Definition 1. *(Threat Model (Blanchard et al., 2017; Chen et al., 2017; Yin et al., 2018)). In the t^{th} iteration, let $\{v_i^t : i \in [m]\}$ be i.i.d. random vectors in $\mathbb{R}^d$, where $v_i^t = \nabla F_i(x^t)$. The set of correct vectors $\{v_i^t : i \in [m]\}$ is partially replaced by arbitrary vectors, which results in $\{\tilde{v}_i^t : i \in [m]\}$, as defined in Equation (1). In other words, a correct gradient is $\nabla F_i(x^t)$, while a Byzantine gradient, marked as "$*$", is assigned arbitrary value. We assume that q out of m vectors are Byzantine, where $2q < m$. Furthermore, the indices of faulty workers can change across different iterations. If the failures are caused by attackers, the threat model includes the case where the attackers can collude.*

The notations used in this paper is summarized in Table 1.

Table 1: Notations

Notation	Description
m	Number of workers
n	Minibatch size on each worker
T	Number of iterations
$[m]$	Set of integers $\{1, \ldots, m\}$
q	Number of Byzantine workers
γ	Learning rate
x	Model parameters
$\tilde{v}_i^t$	Gradient sent by the ith worker in the t^{th} iteration, potentially Byzantine
v_i^t	Correct gradient produced by the ith worker in the t^{th} iteration
$\|\cdot\|$	All the norms in this paper are l_2-norms
$\langle a, b\rangle$	Inner product between a and b

4 DEFENSE TECHNIQUES

In this section, we introduce two popular robust aggregation rules against Byzantine failures in distributed synchronous SGD: coordinate-wise median and Krum. For the remainder of this paper, we ignore the iteration superscript t in $\tilde{v}_i^t$ and v_i^t for convenience.

4.1 COORDINATE-WISE MEDIAN

Definition 2. *(Coordinate-wise Median (Yin et al., 2018)) We define the coordinate-wise median aggregation rule* $\texttt{Median}(\cdot)$ *as*

$$med = \texttt{Median}(\{\tilde{v}_i : i \in [m]\}),$$

where for any $j \in [d]$, the jth dimension of med is $med_j = median\left(\{(\tilde{v}_1)_j, \ldots, (\tilde{v}_m)_j\}\right)$, $(\tilde{v}_i)_j$ is the jth dimension of the vector $\tilde{v}_i$, $median(\cdot)$ is the one-dimensional median.

4.2 KRUM

Definition 3. *(Krum (Blanchard et al., 2017))*

$$\texttt{Krum}(\{\tilde{v}_i : i \in [m]\}) = \tilde{v}_k, \quad k = \operatorname*{argmin}_{i \in [m]} KR(\tilde{v}_i),$$

$$KR(\tilde{v}_i) = \sum_{i \to j} \|\tilde{v}_i - \tilde{v}_j\|^2,$$

where $i \to j$ are the indices of the $m - q - 2$ nearest neighbours of $\tilde{v}_i$ in $\{\tilde{v}_j : j \in [m], i \neq j\}$ as measured by squared Euclidean distance.

For convenience, we refer to the coordinate-wise median and Krum as `Median` and `Krum`.

5 ATTACK TECHNIQUES

In this section, we revise the definition of Byzantine tolerance in distributed synchronous SGD. Then, we theoretically analyze the Byzantine tolerance of coordinate-wise median and Krum, and show that under certain conditions, these two robust aggregation rules are no longer Byzantine-tolerant.

5.1 INNER PRODUCT MANIPULATION

In the previous work on Byzantine-tolerant SGD algorithms, most of the robust aggregation rules only guarantee that the robust estimator is not arbitrarily far away from the mean of the correct gradients. In other words, the distance between the robust estimator and the correct mean is upper-bounded. However, for gradient descent algorithms, to guarantee the descent of the loss, the inner product between the true gradient and the robust estimator must be non-negative:

$$\langle \nabla F(x), \texttt{Aggr}(\{\tilde{v}_i : i \in [m]\})\rangle \geq 0,$$

so that at least the loss will not increase in expectation. In particular, bounded distance is not enough to guarantee robustness, if the attackers manipulate the Byzantine gradients and make the inner product negative.

The intuition underlying the inner product manipulation attack is that, when gradient descent algorithm converges, the gradient $\nabla F(x^t)$ approaches 0. Thus, even if the distance between the robust estimator and the correct mean is bounded, it is still possible to manipulate their inner product to be negative, especially when the upper bound of such distance is large.

We formally define a revised version of Byzantine tolerance for distributed synchronous SGD (DSSGD-Byzantine tolerance):

Definition 4. *(DSSGD-Byzantine Tolerance) Without loss of generality, suppose that in a specific iteration, the server receives $(m - q)$ correct gradients $\mathcal{V} = \{v_1, \ldots, v_{m-q}\}$ and q Byzantine gradients $\mathcal{U} = \{u_1, \ldots, u_q\}$. We assume that the correct gradients have the same expectation $\mathbb{E}[v_i] = g, \forall i \in [m-q]$. An aggregation rule* $\texttt{Aggr}(\cdot)$ *is said to be DSSGD-Byzantine-tolerant if*

$$\langle g, \quad \mathbb{E}\left[\texttt{Aggr}(\mathcal{V} \cup \mathcal{U})\right]\rangle \geq 0.$$

With the revised definition, now we theoretically analyze the DSSGD-Byzantine tolerance of coordinate-wise median and Krum.

Remark 1. *Note that we do not argue that the theoretical guarantees in the previous work are wrong. Instead, our claim is that the theoretical guarantees on the*

bounded distances are not enough to secure distributed synchronous SGD. In particular, DSSGD-Byzantine tolerance is different from the Byzantine tolerance proposed in previous work.

5.2 COORDINATE-WISE MEDIAN

The following theorem shows that under certain conditions, `Median` is not DSSGD-Byzantine-tolerant.

Theorem 1. *We consider the worst case where $m - 2q = 1$. The server receives $(m-q)$ correct gradients $\mathcal{V} = \{v_1, \ldots, v_{m-q}\}$ and q Byzantine gradients $\mathcal{U} = \{u_1, \ldots, u_q\}$. We assume that the stochastic gradients have identical expectation $\mathbb{E}[v_i] = g, \forall i \in [m-q]$, and non-zero coordinate-wise variance $\mathbb{E}[((v_i)_j - g_j)^2] \geq \sigma^2, \forall i \in [m-q], j \in [d]$, where $(v_i)_j$ is the jth coordinate of v_i, and g_j is the jth coordinate of g. When $\max_{j\in[d]} |g_j| < \frac{\sigma}{\sqrt{m-q-1}}$, there exist Byzantine gradients $\mathcal{U} = \{u_1, \ldots, u_q\}$ such that*

$$\langle g, \quad \mathbb{E}\left[\texttt{Median}(\mathcal{V} \cup \mathcal{U})\right] \rangle < 0.$$

Proof. (sketch) Since median is independently taken in each coordinate, it is sufficient to prove Byzantine vulnerability for one coordinate or scalars. Thus, for convenience, with a little bit abuse of notation, we suppose that the correct gradients $\mathcal{V} = \{v_1, \ldots, v_{m-q}\}$ and q Byzantine gradients $\mathcal{U} = \{u_1, \ldots, u_q\}$ are all scalars. We only need to show that under certain attacks, the aggregated value $\texttt{Median}(\mathcal{V} \cup \mathcal{U})$ has a different sign than $\sum_{i\in[m-q]} v_i$.

Without loss of generality, we assume that $g = \frac{1}{m-1}\sum_{i\in[m-q]} \mathbb{E}[v_i] > 0$ (the mirror case can be easily proved with a similar procedure). The Byzantine gradients are all assigned negative value: $u_i < 0, \forall i \in [q]$. Furthermore, we make the Byzantine gradients small enough such that $u_i < \min(\mathcal{V}), \forall i \in [q]$.

By sorting the correct gradients, we can define the sequence $\{v_{1:m-q}, \ldots, v_{m-q:m-q}\}$, where $v_{i:m-q}$ is the ith smallest element in $\{v_1, \ldots, v_{m-q}\}$:

$$v_{1:m-q} \leq v_{2:m-q} \leq \cdots \leq v_{m-q:m-q}.$$

We also define the expectation of the ith smallest element: $\mu_{i:m-q} = \mathbb{E}[v_{i:m-q}]$.

Then, it is easy to check that $\texttt{Median}(\mathcal{V} \cup \mathcal{U}) = v_{1:m-q}$, and $\mathbb{E}\left[\texttt{Median}(\mathcal{V} \cup \mathcal{U})\right] = \mu_{1:m-q}$.

Using Theorem 1(b) from Hawkins (1971) (equiv. 9(a) from Arnold et al. (1979)), we have

$$\mu_{1:m-q} \leq g - \frac{\sigma}{\sqrt{m-q-1}}.$$

Thus, when $g < \frac{\sigma}{\sqrt{m-q-1}}$, $\mathbb{E}\left[\texttt{Median}(\mathcal{V} \cup \mathcal{U})\right]$ is negative. □

Remark 2. *When gradient descent converges, the expectation of the gradient g approaches 0. Furthermore, since the gradient produced by the correct workers are stochastic, the variance is non-zero. Thus, eventually, the condition $\max_{j\in[d]} |g_j| < \frac{\sigma}{\sqrt{m-q-1}}$ will be satisfied. To make things worse, the closer SGD approaches a critical point, the less likely the coordinate-wise median is DSSGD-Byzantine-tolerant.*

Remark 3. *The proof of Theorem 1 provides the intuition for constructing adversarial gradients for the attackers. In practice, in each coordinate, the attackers only need to guarantee that all the Byzantine values are much smaller than the smallest correct value if the correct expectation is positive, or much larger than the largest correct value if the correct expectation is negative. Then, hopefully (for the attackers), if the variance is large enough, the smallest/largest value has the opposite sign to the correct expectation. Hence, the attackers can successfully manipulate the aggregated value into the opposite direction to the correct expectation.*

5.2.1 Toy Example

We provide an 1-dimensional toy example to illustrate how easily `Median` can fail. Suppose there are 3 correct gradients $\mathcal{V} = \{-0.1, 0.1, 0.3\}$ with the mean 0.1, and 2 Byzantine gradient $\mathcal{U} = \{-4, -2\}$ with the negative mean -3. According to Definition 2, it is easy to check that $\texttt{Median}(\mathcal{U} \cap \mathcal{V}) = -0.1$, which means that `Median` produces a value with the opposite sign of the mean of the correct gradients.

5.3 KRUM

The following theorem proves that under certain conditions, Krum is not DSSGD-Byzantine-tolerant. Note that Krum requires that $m - 2q > 2$.

Theorem 2. *We consider the worst case where $m - 2q = 3$. The server receives $(m-q)$ correct gradients $\mathcal{V} = \{v_1, \ldots, v_{m-q}\}$ and q Byzantine gradients $\mathcal{U} = \{u_1, \ldots, u_q\}$. We assume that the stochastic gradients have identical expectation $\mathbb{E}[v_i] = g, \forall i \in [m-q]$. We define the mean of the correct gradients $\bar{v} = \frac{1}{m-q}\sum_{i\in[m-q]} v_i$. We assume that the correct gradients are bounded by $\|v_i - \bar{v}\|^2 \leq \|\bar{v}\|^2, \forall i \in [m-q]$. Furthermore, we assume that $v_i \neq v_j, \forall i \neq j, i, j \in [m-q]$, and $\exists \beta$ such that $\|v_i - v_j\|^2 \geq \beta^2, \forall i \neq j, i, j \in [m-q]$. We take $u_1 = u_2 = \cdots = u_q = -\epsilon\bar{v}$, where ϵ is a small positive constant value such that $\epsilon^2\|\bar{v}\|^2 \leq \beta^2$. When $(m-q)$ is large enough: $m - q > \frac{2(\epsilon+2)^2}{\epsilon^2} + 2$, we have*

$$\langle g, \quad \mathbb{E}\left[\texttt{Krum}(\mathcal{V} \cup \mathcal{U})\right] \rangle < 0.$$

Proof. (sketch) For $\forall u \in \mathcal{U}$, $u = -\epsilon\bar{v}$, where $\bar{v} = \frac{1}{m-1}\sum_{i\in[m-q]} v_i$.

Since any $u \in \mathcal{U}$ is identical, the nearest $(m-q-4)$ neighbours of u must belong to $\mathcal{U}$. The remaining $(m-q-2)-(m-q-4)=2$ nearest neighbours must belong to the set of correct gradients $\mathcal{V}$. Thus, we have

$$KR(u) \leq 2\|\bar{v}+\bar{v}+\epsilon\bar{v}\|^2 = 2(\epsilon+2)^2\|\bar{v}\|^2.$$

For the correct gradients $\forall v \in \mathcal{V}$, there are two cases:

- **Case 1:** There are some $u \in \mathcal{U}$ which belong to the $(m-q-2)$ nearest neighbours of v.

  Suppose there are a_1 nearest neighbours in $\mathcal{V}$ and a_2 nearest neighbours in $\mathcal{U}$, where $a_1+a_2 = m-q-2$. Since the correct gradients are bounded by $\|v_i-\bar{v}\|^2 \leq \|\bar{v}\|^2, \forall i \in [m-q]$, it is easy to check that $\|v-u\|^2 \geq \epsilon^2\|\bar{v}\|^2$. Thus, we have

  $$KR(v) \geq a_1\beta^2 + a_2\|v-u\|^2 \geq (m-q-2)\epsilon^2\|\bar{v}\|^2.$$

- **Case 2:** There are no $u \in \mathcal{U}$ which belong to the $(m-q-2)$ nearest neighbours of v. Thus, we have

  $$KR(v) \geq (m-q-2)\beta^2 \geq (m-q-2)\epsilon^2\|\bar{v}\|^2.$$

In both cases, we have $KR(v) \geq (m-q-2)\epsilon^2\|\bar{v}\|^2$. Thus, when $(m-q)$ is large enough: $m-q > \frac{2(\epsilon+2)^2}{\epsilon^2}+2$, we have

$$\begin{aligned} KR(u) &\leq 2(\epsilon+2)^2\|\bar{v}\|^2 < (m-q-2)\epsilon^2\|\bar{v}\|^2 \\ &\leq KR(v). \end{aligned}$$

As a result, $\texttt{Krum}(\mathcal{V}\cap\mathcal{U}) = u = -\epsilon\bar{v}$. Thus, $\mathbb{E}\left[\texttt{Krum}(\mathcal{V}\cap\mathcal{U})\right] = -\epsilon g$.

□

Remark 4. *In the theorem above, we assume that all the correct gradients are inside a Euclidean ball centered at their mean: $\|v_i-\bar{v}\|^2 \leq \|\bar{v}\|^2, \forall i \in [m-q]$. Such assumption can not always be satisfied, but it is reasonable that the random samples are sometimes inside such a Euclidean ball, if the variance is not too large. On the other hand, we assume that the pair-wise distances between the correct gradients are lower-bounded by $\beta > 0$. Almost surely, such β exists, no matter how small it is. Note that the Byzantine attackers are supposed to be omniscient. Thus, the attackers can spy on the honest workers, and obtain $\mathcal{V}$ and β. Then, the attackers can choose an ϵ such that $\epsilon^2\|\bar{v}\|^2 \leq \beta^2$. Finally, we only need the number of workers to be large enough, so that $m-q > \frac{2(\epsilon+2)^2}{\epsilon^2}+2$.*

Remark 5. *The proof of Theorem 2 provides the intuition ofor constructing adversarial gradients for the attackers. In practice, the attackers only need to assign $\frac{\epsilon}{m-q}\sum_{i\in[m-q]} v_i$ to all the Byzantine gradients, with an $\epsilon > 0$ small enough.*

Remark 6. *`Krum` (Blanchard et al., 2017) requires the assumption that $c\sigma < \|g\|$ for convergence, where c is a general constant, σ is the maximal variance of the gradients, and g is the gradient in expectation. Note that $\|g\| \to 0$ when SGD converges to a critical point. Thus, such an assumption is never guaranteed to be satisfied if the variance is non-zero. Furthermore, the better SGD converges, the less likely such an assumption can be satisfied.*

5.3.1 Toy Example

Note that the assumptions made in Theorem 2 are sufficient but not necessary conditions for the DSSGD-Byzantine vulnerability of `Krum`. In practice, it can be easier to find an ϵ that crashes `Krum`, especially for 1-dimensional cases.

We provide an 1-dimensional toy example to show how easily `Krum` can fail. Suppose there are 6 correct gradients $\mathcal{V} = \{0, 0.02, 0.14, 0.26, 0.38, 0.5\}$ with the mean 0.2167, and 3 Byzantine gradient $\mathcal{U} = \{-0.1, -0.1, -0.1\}$ with the negative mean -0.1. According to Definition 3, the corresponding function values $KR(\cdot)$ of $\mathcal{U}\cap\mathcal{V} = \{-0.1, -0.1, -0.1, 0, 0.02, 0.14, 0.26, 0.38, 0.5\}$ are $\{0.0244, 0.0244, 0.0244, 0.0304, 0.0436, 0.1060, 0.1440, 0.2160, 0.4320\}$. Thus, $\texttt{Krum}(\mathcal{U}\cap\mathcal{V}) = -0.1$, which means that `Krum` chooses the Byzantine gradient with the opposite sign of the mean of the correct gradients.

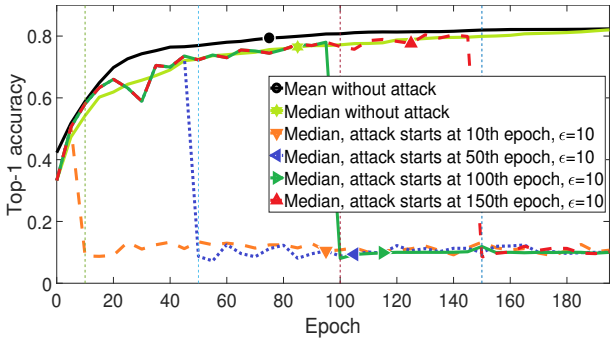
6 CASE STUDY

In this section, we implement special attack strategies for `Median` and `Krum`, and evaluate our attack strategies on a real-world application. The attack strategies are designed based on intuition underlying Theorem 1 and Theorem 2, as discussed in Remark 3 and Remark 5.

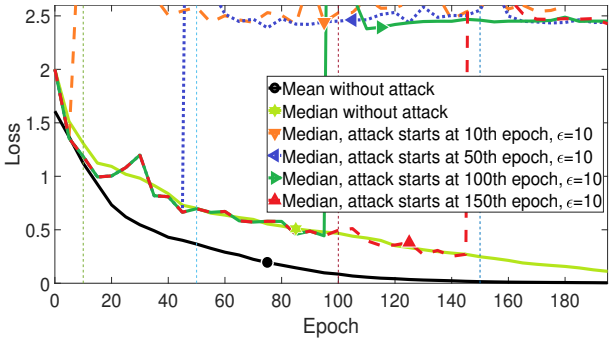
6.1 DATASETS AND EVALUATION METRICS

We conduct experiments on the benchmark CIFAR-10 image classification dataset (Krizhevsky & Hinton, 2009), which is composed of 50k images for training and 10k images for testing. We use a convolutional neural network (CNN) with 4 convolutional layers followed by 1 fully connected layer, implemented by MXNet (Chen et al., 2015). The detailed network architecture can be found in the appendix. For any worker, the minibatch size for SGD is 50.

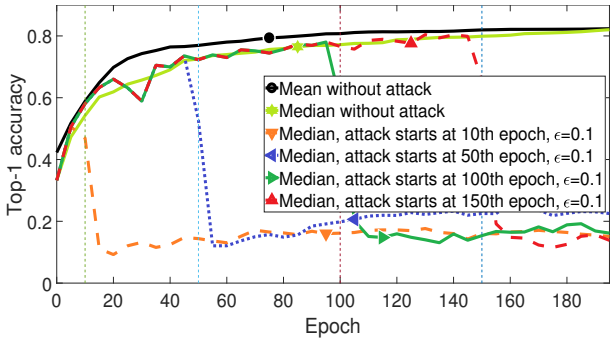
In each experiment, we launch 25 worker processes. We repeat each experiment 10 times and take the average. We use top-1 accuracy on the testing set and the cross-entropy loss function on the training set as the evaluation metrics.



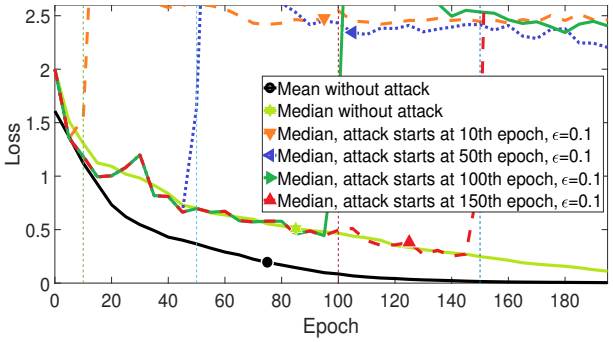
(a) Top-1 Accuracy on Testing Set, $\epsilon = 10$



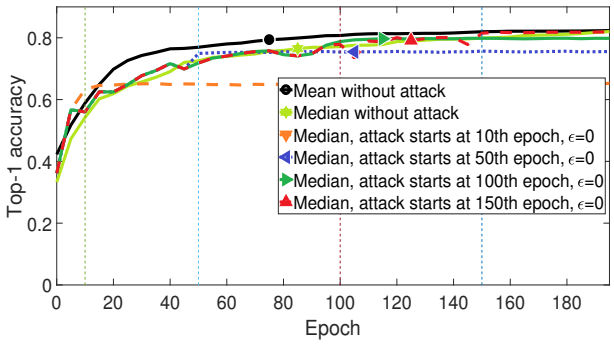
(b) Cross Entropy on Training Set, $\epsilon = 10$



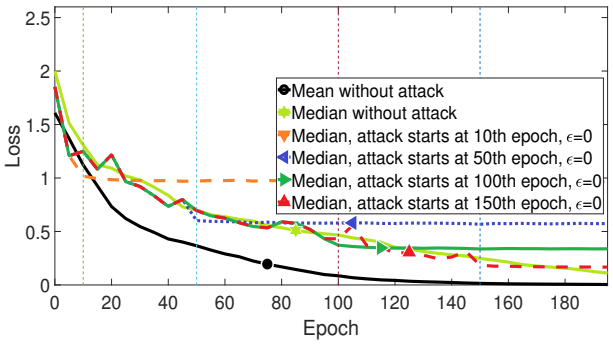
(c) Top-1 Accuracy on Testing Set, $\epsilon = 0.1$



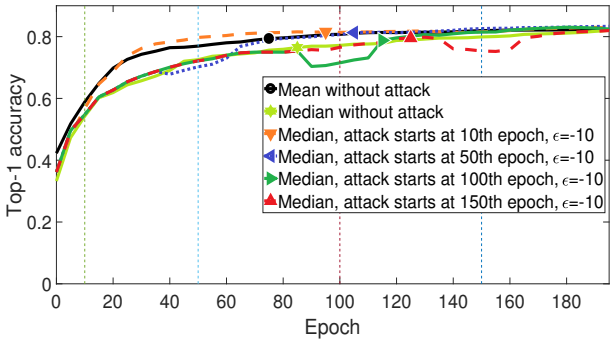
(d) Cross Entropy on Training Set, $\epsilon = 0.1$



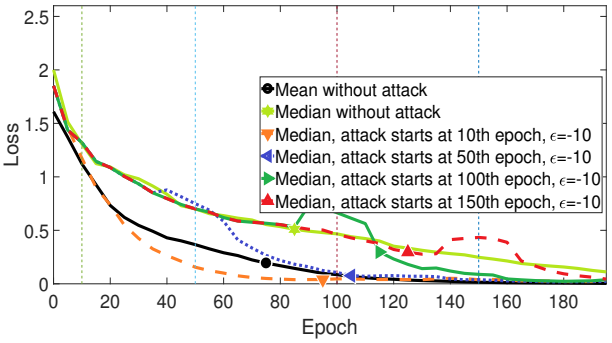
(e) Top-1 Accuracy on Testing Set, $\epsilon = 0$



(f) Cross Entropy on Training Set, $\epsilon = 0$

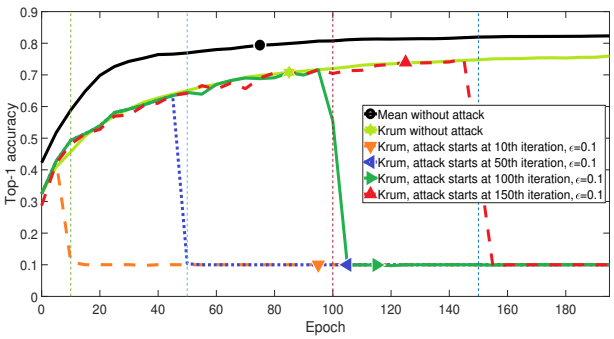


(g) Top-1 Accuracy on Testing Set, $\epsilon = -10$

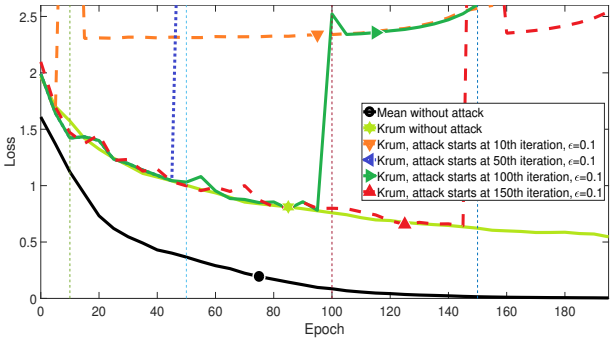


(h) Cross Entropy on Training Set, $\epsilon = -10$

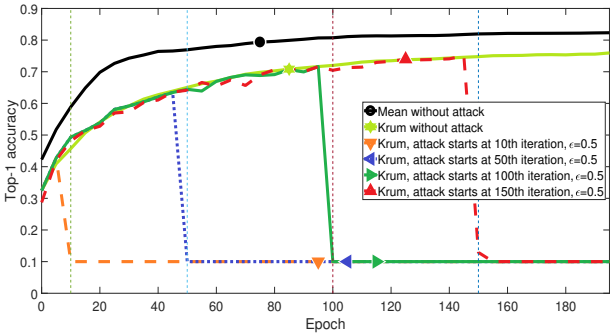
Figure 4: Convergence on training set, using `Median` as aggregation rule. $\epsilon \in \{10, 0.1, 0, -10\}$.



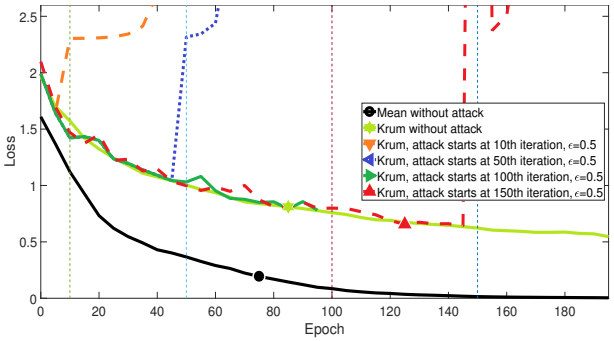
(a) Top-1 Accuracy on Testing Set, $\epsilon = 0.1$



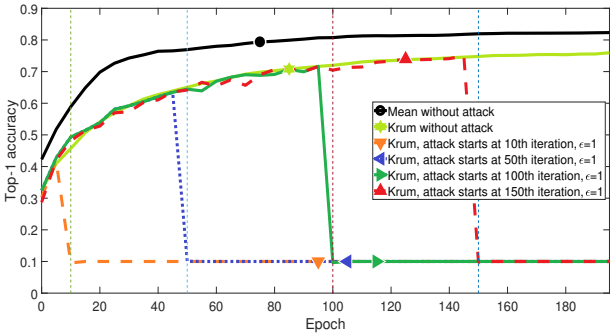
(b) Cross Entropy on Training Set, $\epsilon = 0.1$



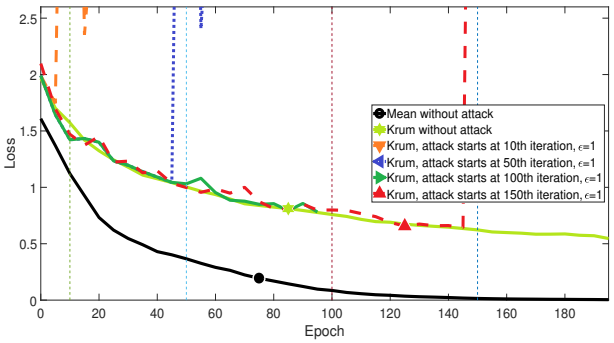
(c) Top-1 Accuracy on Testing Set, $\epsilon = 0.5$



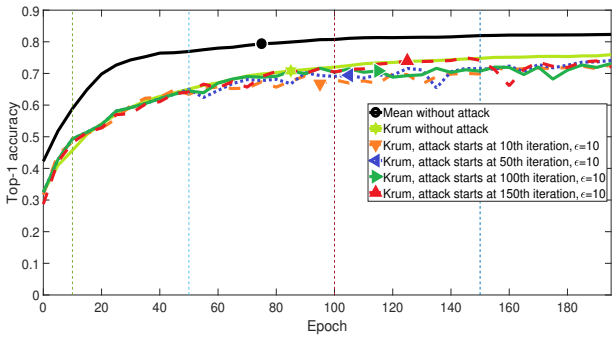
(d) Cross Entropy on Training Set, $\epsilon = 0.5$



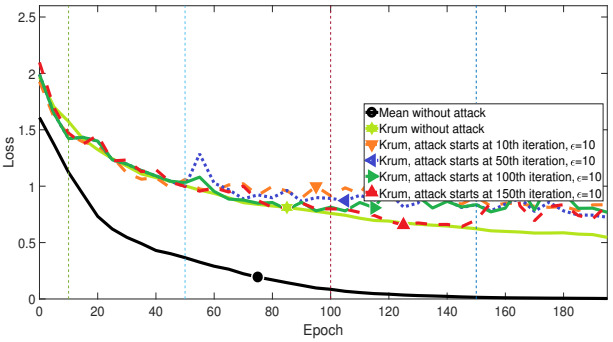
(e) Top-1 Accuracy on Testing Set, $\epsilon = 1$



(f) Cross Entropy on Training Set, $\epsilon = 1$



(g) Top-1 Accuracy on Testing Set, $\epsilon = 10$



(h) Cross Entropy on Training Set, $\epsilon = 10$

Figure 5: Convergence on training set, using `Krum` as aggregation rule. $\epsilon \in \{0.1, 0.5, 1, 10\}$.

We use the averaging, `Median`, and `Krum` without attacks as the gold standards, which are referred to as `Mean without attack`, `Median without attack`, and `Krum without attack`. We start the attack at different epochs, so that SGD can warm up and make some progress first. We include some additional experiments in the appendix.

6.2 MEDIAN

In each iteration, the server receives $m = 25$ gradients. A randomly selected subset of $q = 12$ correct gradients are replaced by Byzantine gradients. We define the set of Byzantine gradients as $\mathcal{U} = \{u_1, \ldots, u_{12}\}$, and the set of the remaining correct gradients as $\mathcal{V} = \{v_1, \ldots, v_{13}\}$. Our attack strategy is as follows:

$$u_1 = u_2 = \cdots = u_{12} = -\frac{\epsilon}{13}\sum_{i=1}^{13} v_i.$$

According to Theorem 1 and Remark 3, `Median` is vulnerable to positive ϵ with large magnitude $|\epsilon|$.

We test the above attack strategy with different ϵ. The results are shown in Figure 4. `Median` fails when $\epsilon > 0$. When $\epsilon = 0$, `Median` gets stuck and stops making progress. When $\epsilon < 0$, `Median` successfully defends against the attack.

6.3 KRUM

In each iteration, the server receives $m = 25$ gradients. A randomly selected subset of $q = 11$ correct gradients are replaced by Byzantine gradients. We define the set of Byzantine gradients as $\mathcal{U} = \{u_1, \ldots, u_{11}\}$, and the set of the remaining correct gradients as $\mathcal{V} = \{v_1, \ldots, v_{14}\}$. Our attack strategy is as follows:

$$u_1 = u_2 = \cdots = u_{11} = -\frac{\epsilon}{14}\sum_{i=1}^{14} v_i.$$

According to Theorem 2 and Remark 5, `Krum` is vulnerable to positive ϵ with small magnitude $|\epsilon|$.

We test the above attack strategy with different ϵ. The results are shown in Figure 5. `Krum` fails when $\epsilon > 0$ is small. When ϵ is large enough, `Krum` successfully defends against the attack.

6.4 DISCUSSION

Surprisingly, both `Median` and `Krum` are more vulnerable than we expected. Note that our theorems only analyze the worst cases. There are other cases where `Median` and `Krum` can fail.

For `Median`, even if we take $\epsilon = 0$, SGD still performs badly. Theoretically, even if we do not use positive ϵ, small ϵ can still enlarge the variance of SGD, which can be potentially harmful to the convergence. We can see that with large negative ϵ, the defense of `Median` is successful. Our experiments reveal certain new vulnerabilities of `Median` in distributed synchronous SGD. The experiments conducted by Yin et al. (2018) do not fail because the attacker only changes the labels of the poisoned training data by flipping a $label \in \{0, \ldots, 9\}$ to $9 - label$. It is very likely that such an attack produces Byzantine gradients surrounding the correct gradients coordinate-wisely on both sides. However, according to Theorem 1 and Remark 3, an effective attack should place the Byzantine gradient on one and only one side of the correct gradients, which is the side opposite to the mean of the correct gradients, coordinate-wise.

For `Krum`, small positive ϵ makes SGD vulnerable. Furthermore, even if we take $\epsilon = 1$, `Krum` still fails. Our experiments reveal certain new vulnerabilities of `Krum` in distributed synchronous SGD. The experiments conducted by Blanchard et al. (2017) do not fail even though a similar attack strategy called "omniscient" is conducted. The reason is that, in the paper of Blanchard et al. (2017), the attacker "proposes the opposite vector, scaled to a large length", which is similar to our attack strategy with a large ϵ.

Guided by our theoretical analysis, we designed efficient attack strategies for both `Median` and `Krum`. Our results show that the definition of Byzantine tolerance for distributed synchronous SGD should be revised. Using our definition of DSSGD-Byzantine tolerance, research can be conducted to design better defense techniques.

7 CONCLUSION

We propose a revised definition of Byzantine tolerance for distributed synchronous SGD. With the new definition, we theoretically and empirically examine the Byzantine tolerance of two prevailing robust aggregation rules. Guided by our theoretical analysis, attack techniques can be designed to fail the aggregation rules. In the future, we hope new defense techniques can be designed using our revised definition of Byzantine tolerance.

Acknowledgements

This work was funded in part by NSF CNS 1409416, by a gift from Microsoft, and by computational resources donated by Intel, AWS, and Microsoft Azure.

References

Alistarh, D., Allen-Zhu, Z., and Li, J. Byzantine Stochastic Gradient Descent. *arXiv preprint arXiv:1803.08917*, 2018.

Arnold, B. C., Groeneveld, R. A., et al. Bounds on expectations of linear systematic statistics based on dependent samples. *The Annals of Statistics*, 7(1):220–223, 1979.

Athalye, A., Carlini, N., and Wagner, D. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *International Conference on Machine Learning*, pp. 274–283, 2018.

Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing*, 1(1):11–33, 2004.

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018.

Bhagoji, A. N., Chakraborty, S., Mittal, P., and Calo, S. Analyzing Federated Learning through an Adversarial Lens. *arXiv preprint arXiv:1811.12470*, 2018.

Blanchard, P., Guerraoui, R., Stainer, J., et al. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Advances in Neural Information Processing Systems*, pp. 118–128, 2017.

Chen, L., Wang, H., Charles, Z., and Papailiopoulos, D. DRACO: Byzantine-resilient Distributed Training via Redundant Gradients. In *International Conference on Machine Learning*, pp. 902–911, 2018.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

Chen, Y., Su, L., and Xu, J. Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent. *POMACS*, 1:44:1–44:25, 2017.

Feng, J., Xu, H., and Mannor, S. Distributed robust learning. *arXiv preprint arXiv:1409.5937*, 2014.

Fischer, M. J., Lynch, N. A., and Paterson, M. S. Impossibility of distributed consensus with one faulty process. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE, 1982.

Guerraoui, R., Rouault, S., et al. The Hidden Vulnerability of Distributed Learning in Byzantium. In *International Conference on Machine Learning*, pp. 3518–3527, 2018.

Hawkins, D. M. On the bounds of the range of order statistics. *Journal of the American Statistical Association*, 66(335):644–645, 1971.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

Lamport, L., Shostak, R. E., and Pease, M. C. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.*, 4:382–401, 1982.

Li, M., Andersen, D. G., Park, J. W., Smola, A. J., Ahmed, A., Josifovski, V., Long, J., Shekita, E. J., and Su, B.-Y. Scaling Distributed Machine Learning with the Parameter Server. In *OSDI*, volume 14, pp. 583–598, 2014a.

Li, M., Andersen, D. G., Smola, A. J., and Yu, K. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pp. 19–27, 2014b.

Lynch, N. A. *Distributed algorithms*. Elsevier, 1996.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

Su, L. and Vaidya, N. H. Fault-Tolerant Multi-Agent Optimization: Optimal Iterative Distributed Algorithms. In *PODC*, 2016a.

Su, L. and Vaidya, N. H. Defending non-Bayesian learning against adversarial attacks. *arXiv preprint arXiv:1606.08883*, 2016b.

Tanenbaum, A. S. and Van Steen, M. *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

Xie, C., Koyejo, S., and Gupta, I. SLSGD: Secure and Efficient Distributed On-device Machine Learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2019a.

Xie, C., Koyejo, S., and Gupta, I. Zeno: Distributed Stochastic Gradient Descent with Suspicion-based Fault-tolerance. In *International Conference on Machine Learning*, pp. 6893–6901, 2019b.

Xing, E. P., Ho, Q., Xie, P., and Wei, D. Strategies and principles of distributed machine learning on big data. *Engineering*, 2(2):179–195, 2016.

Yin, D., Chen, Y., Ramchandran, K., and Bartlett, P. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. *arXiv preprint arXiv:1803.01498*, 2018.