

---

# Distributed Momentum for Byzantine-resilient Learning

---

El-Mahdi El-Mhamdi<sup>1</sup> Rachid Guerraoui<sup>1</sup> Sébastien Rouault<sup>1</sup>

## Abstract

Momentum is a variant of gradient descent that has been proposed for its benefits on convergence. In a distributed setting, momentum can be implemented either at the server or the worker side. When the aggregation rule used by the server is linear, commutativity with addition makes both deployments equivalent. Robustness and privacy are however among motivations to abandon linear aggregation rules. In this work, we demonstrate the benefits on robustness of using momentum at the worker side. We first prove that computing momentum at the workers reduces the variance-norm ratio of the gradient estimation at the server, strengthening Byzantine resilient aggregation rules. We then provide an extensive experimental demonstration of the robustness effect of worker-side momentum on distributed SGD.

## 1. Introduction

Gradient descent is the driving force of the recent successes in machine learning. Large-scale deployment of gradient descent relies on two ideas: stochastic approximation and distribution. Stochastic approximation (drastically) reduces the computation time, at the price of introducing variance in the gradient estimations. Distribution alleviates the workload on a single machine but, as we discuss below, the multiplicity of elements inevitably increases the likelihood of (malicious) faults.

In the distributed parameter server setting, the training of a model is basically performed as follows. A central machine, called the *server*, sends the current model (the vector of parameters) to other machines, called *workers*. These use their share of data (either their local and private data, or data provided by the server for training purpose) to compute a gradient estimate which is in turn sent to the server. As the server receives the gradients from different workers, the

---

Author list in alphabetical order. <sup>1</sup>Distributed Computing Laboratory (DCL), École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. Correspondence to: Sébastien Rouault <sebastien.rouault@epfl.ch>.

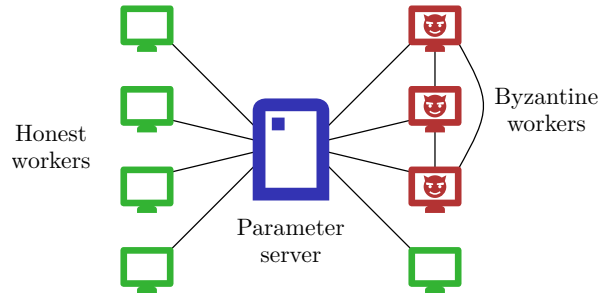


Figure 1. A parameter server setup with  $n = 8$  workers, among which  $f = 3$  are *Byzantine* (i.e., adversarial) workers. A black line represents a bidirectional communication channel.

server typically averages their values to update the model if the setting is synchronous, or updates the model as individual gradients are received if the model is asynchronous.

When all the workers are reliable and provide correct estimates of the gradient, this setting has close to optimal behavior (Lian et al., 2015; Zhang et al., 2016; Dean et al., 2012). Many practical factors could however make the correctness assumption of the workers doubtful. These factors span a large spectrum of causes, from software bugs, noisy or poisonous data, stale machines or worse, malicious attackers controlling some machines.

The Byzantine abstraction is a very general fault model in distributed systems (Lamport et al., 1982). The standard, golden solution for Byzantine fault tolerance is the *state machine replication* approach (Schneider, 1990). This approach is however based on *replication*, which is unsuitable for distributed machine learning and stochastic gradient descent, such as federated learning (Konečný et al., 2015). Workers could be independent entities, who could not be replicated for obvious privacy, scalability or legal reasons. For instance, in the context of federated learning, recent work has shown that Byzantine fault tolerance serves as a good basis to study poisoning (Bagdasaryan et al., 2018; Sun et al., 2019). In that same context, recent results show that Byzantine-resilient aggregation rules are effective against *distributed backdoor attacks* (Xie et al., 2020).

The key vulnerability in the standard parameter server rests upon how gradients are aggregated. Since 2017, many alternatives to averaging have been proposed ((Alistarh et al.,

2018; Chen et al., 2018; Yin et al., 2018; Xie et al., 2018a; Blanchard et al., 2017; El-Mhamdi et al., 2018; Damaskinos et al., 2018; Yang & Bajwa, 2019b; TianXiang et al., 2019; Bernstein et al., 2019; Xie et al., 2018a; Yang & Bajwa, 2019a; Chen et al., 2018; Xie et al., 2018b; Yang et al., 2019; Rajput et al., 2019; Muñoz-González et al., 2019) to list a few). In synchronous settings, these solutions consist in replacing the averaging of gradients by a robust alternative such as the median and its variants (Blanchard et al., 2017; El-Mhamdi et al., 2018; Xie et al., 2018a) or redundancy schemes (Chen et al., 2018; Rajput et al., 2019). In asynchronous settings, since no aggregation can be made, gradients are (ideally) used individually as they are delivered, the robust alternatives are less diverse and are mostly consisting of a filtering scheme (Damaskinos et al., 2018).

One common aspect underlying these methods is their reliance on “quality gradients” from the non-Byzantine workers. Technically: the variance between non-Byzantine gradient estimates must be bounded above a factor of their average norm. This requirement is not new in machine learning (Bottou, 1998), and is actually independent from Byzantine considerations as an unbounded *variance-norm ratio* would prevent convergence.

Is there a way to guarantee “quality gradient” at the non-Byzantine workers? Addressing this question is crucial to put Byzantine-resilient gradient descent to work.

We provide a positive answer to this question by using momentum (Rumelhart et al., 1986). Momentum consists in summing a series of past gradients with the new one using an exponential decay factor  $\mu$  ( $0 < \mu < 1$ ), instead of using the new gradient alone. Momentum can be computed at the server side, when the update is performed, or at the workers’ side, when gradients are still computed (Lin et al., 2018). In non-Byzantine-resilient settings, both deployments are equivalent, as the gradient aggregation used at the server is linear and commutes with addition. In practice, momentum is typically employed at the server side. In this work, we propose to use momentum at the workers’ side since none of the existing Byzantine-resilient aggregation rules is linear.

We first show theoretically that indeed we can guarantee “quality gradient” by using momentum at the workers. Then we report on an extensive experimental assessment of this claim. In particular, and while using momentum at the workers has *no additional overhead* over momentum at the server, this technique led to an observed  $\times 11$  reduction on cross-accuracy drop due to Byzantine actors (Section 4.3).

**Contributions.** Essentially, we show for the first time that applying momentum at the workers significantly boosts robustness against Byzantine behavior. We prove that computing momentum at the workers reduces the *variance-norm ratio* of the honest gradient estimations at the server, a key

quantity for any robust alternative to averaging which approximates a high-dimensional median; for instance (Blanchard et al., 2017; Xie et al., 2018a; El-Mhamdi et al., 2018). In particular, we show that combining now-standard defense mechanisms (Blanchard et al., 2017; Xie et al., 2018a; El-Mhamdi et al., 2018) with momentum (at the worker side) ensures previously unavailable safety guarantees and counters state-of-the-art attacks such as (Baruch et al., 2019; Xie et al., 2019). We report on an extensive experimental evaluation of this claim with 88 different tested sets of hyperparameters (440 trained models in total), spanning the 2 mentioned state-of-the-art attacks and 3 defenses.

**Paper Organization.** Section 2 formalizes the problem and provides the necessary background. Section 3 presents our theoretical contribution and compares the usage of momentum at the workers versus at the server. Section 4 describes our experimental settings in details, before presenting and analysing our experimental results. Section 5 discusses related and future work.

Due to space limitation, only a representative fraction of the experimental results is presented in the main paper. The supplementary material reports on the entirety of our experiments, along with the code and procedure to reproduce all of our results (including the graphs).

## 2. Background

### 2.1. Byzantine Distributed SGD

**Stochastic Gradient Descent (SGD).** We consider the classical problem of optimizing a non-convex, differentiable loss function  $Q : \mathbb{R}^d \rightarrow \mathbb{R}$ , where  $Q(\theta_t) \triangleq \mathbb{E}_{x \sim \mathcal{D}} [q(\theta_t, x)]$  for a fixed data distribution  $\mathcal{D}$ . Namely, we seek a  $\theta^* \in \mathbb{R}^d$  such that:  $\nabla Q(\theta^*) = 0$  (1)

Using SGD, we initially pick a random  $\theta_0 \in \mathbb{R}^d$ . Then at every step  $t \geq 0$ , we uniformly sample  $b$  datapoints  $x_1 \dots x_b$  from  $\mathcal{D}$  to estimate a *gradient*  $g_t \triangleq \frac{1}{b} \sum_{k=1}^b \nabla q(\theta_t, x_k) \approx \nabla Q(\theta_t)$ . Finally, for step  $t$ , we update the parameter vector using  $\theta_{t+1} = \theta_t - \eta_t g_t$ , where  $\eta_t > 0$  is the *learning rate*.

One field-tested amendment to this update rule is *momentum* (Rumelhart et al., 1986), where each gradient has an exponentially-decreasing effect on every subsequent update. Formally:  $\theta_{t+1} = \theta_t - \eta_t \sum_{u=0}^t \mu^{t-u} g_u$ , with  $0 < \mu < 1$ .

**Distributed SGD with Byzantine workers.** We follow the parameter server model (Li et al., 2014): 1 process (the *parameter server*) holding the parameter vector  $\theta_t \in \mathbb{R}^d$ , and  $n$  other processes (the *workers*) estimating gradients. Among these  $n$  workers, up to  $f < n$  are said *Byzantine*, i.e., adversarial. Unlike the other  $n - f$  *honest* workers, these  $f$  *Byzantine* workers can send arbitrary gradients (Figure 1).

At each step  $t$ , the parameter server receives  $n$  different gradients  $g_t^{(1)} \dots g_t^{(n)}$ , among which  $f$  are arbitrary (sent by the Byzantine workers). So the update equation becomes:

$$\theta_{t+1} = \theta_t - \eta_t G_t$$

$$\text{where: } G_t \triangleq \sum_{u=0}^t \mu^{t-u} F(g_u^{(1)}, \dots, g_u^{(n)}) \quad (2)$$

$$\text{and where: } F : (\mathbb{R}^d)^n \rightarrow \mathbb{R}^d$$

The function  $F$  is called a *Gradient Aggregation Rule* (GAR). If we assume no Byzantine worker, averaging is sufficient; formally:  $F(g_t^{(1)}, \dots, g_t^{(n)}) = \frac{1}{n} \sum_{i=1}^n g_t^{(i)}$ . In the presence of Byzantine workers, a more complex aggregation is performed with a *Byzantine-resilient* GAR. Section 2.2 presents the three Byzantine-resilient GARs studied in this paper, along with their own theoretical requirements.

**Adversarial Model.** The goal of the adversary is to impede the learning process, which can generally be defined as the maximization of the loss  $Q$  or, more judiciously in the image classification tasks used in this paper, as the minimization<sup>1</sup> of the model’s top-1 cross-accuracy.

The adversary cannot directly overwrite  $\theta_t$  at the parameter server. The adversary only submits  $f$  arbitrary gradients to the server per step, via the  $f$  Byzantine workers it controls<sup>2</sup>.

We assume an omniscient adversary. In particular, the adversary knows the GAR used by the parameter server and can generate Byzantine gradients dependent on the honest gradients submitted at the same step.

## 2.2. Byzantine-resilient GARs

We formally present below the 3 GARs studied in this paper.

These GARs are *Byzantine-resilience*, a notion first introduced by (Blanchard et al., 2017) under the name  $(\alpha, f)$ -Byzantine-resilience. When used within its operating assumptions, a Byzantine-resilient GAR guarantees convergence (in the sense of (1)) even in an adversarial setting.

**Definition 1.** Let  $(\alpha, f) \in [0, \frac{\pi}{2}] \times [0, n]$ , with  $n$  the total number of workers. Let  $(g_t^{(1)} \dots g_t^{(n)}) \in (\mathbb{R}^d)^n$ , among which  $n - f$  are independent (“honest”) vectors following the same distribution  $\mathcal{G}_t$ ; the  $f$  other vectors are arbitrary, each possibly dependent on  $\mathcal{G}_t$  and the “honest” vectors.

A GAR  $F$  is said to be  $(\alpha, f)$ -Byzantine resilient iff:

$$g_t \triangleq F(g_t^{(1)}, \dots, g_t^{(n)})$$

satisfies:

<sup>1</sup>I.e., with 10 classes, the worst possible final accuracy is 0.1.

<sup>2</sup>Said otherwise, the  $f$  Byzantine workers can collude.

1.  $\langle \mathbb{E} g_t, \mathbb{E} \mathcal{G}_t \rangle \geq (1 - \sin \alpha) \cdot \|\mathbb{E} \mathcal{G}_t\| > 0$
2.  $\forall r \in \{2, 3, 4\}$ ,  $\mathbb{E} \|g_t\|^r$  is bounded above by a linear combination of the terms  $\mathbb{E} \|\mathcal{G}_t\|^{r_1} \dots \mathbb{E} \|\mathcal{G}_t\|^{r_k}$ , with  $(k, r_1 \dots r_k) \in (\mathbb{N}^*)^{k+1}$  and  $r_1 + \dots + r_k = r$ .

### 2.2.1. KRUM (BLANCHARD ET AL., 2017)

Let  $(f, m) \in \mathbb{N}^2$ , with  $n \geq 2f + 3$  and  $1 \leq m \leq n - f - 2$ .

*Krum* works by assigning a score to each input gradient. The score of  $g_t^{(i)}$  is the sum of the distances between  $g_t^{(i)}$  and its  $n - f - 2$  closest neighbor gradients. *Krum* outputs the arithmetic mean of the  $m$  smallest-scoring gradients<sup>3</sup>.

In our experiments, we set  $m$  to its maximum:  $n - f - 2$ .

To be proven  $(\alpha, f)$ -Byzantine resilient, besides the standard convergence conditions in non-convex optimization (Bottou, 1998), *Krum* requires the honest gradients’ variance  $\mathbb{E} \|\mathcal{G}_t - \mathbb{E} \mathcal{G}_t\|^2$  to be bounded above as follows:

$$2 \cdot \kappa(n, f) \cdot \mathbb{E} \|\mathcal{G}_t - \mathbb{E} \mathcal{G}_t\|^2 < \|\mathbb{E} \mathcal{G}_t\|^2 \quad (3)$$

$$\text{with: } \kappa(n, f) \triangleq n - f + \frac{f(n - f - 2) + f^2(n - f - 1)}{n - 2f - 2}$$

### 2.2.2. MEDIAN (XIE ET AL., 2018A)

Let  $f \in \mathbb{N}$  with  $n \geq 2f + 1$ .

*Median* computes the coordinate-wise median of the input gradients  $g_t^{(1)} \dots g_t^{(n)}$ . Formally for the real-valued median:

$$\text{median}(x_1 \dots x_n) \triangleq \arg \min_{x \in \mathbb{R}} \sum_{i=1}^n |x_i - x|$$

And so, formally for the coordinate-wise *Median*:

$$\text{Median}(g_t^{(1)} \dots g_t^{(n)}) \triangleq \begin{pmatrix} \text{median}(g_t^{(1)[1]} \dots g_t^{(n)[1]}) \\ \vdots \\ \text{median}(g_t^{(1)[d]} \dots g_t^{(n)[d]}) \end{pmatrix}$$

The condition of  $(\alpha, f)$ -Byzantine resilience is:

$$(n - f) \cdot \mathbb{E} \|\mathcal{G}_t - \mathbb{E} \mathcal{G}_t\|^2 < \|\mathbb{E} \mathcal{G}_t\|^2 \quad (4)$$

### 2.2.3. BULYAN (EL-MHAMDI ET AL., 2018)

*Bulyan* uses another Byzantine-resilient GAR to aggregate the input gradients. In the remaining of this paper we will consider *Bulyan* of *Krum*, that we will simply call *Bulyan*.

Let  $(f, m) \in \mathbb{N}^2$ , with  $n \geq 4f + 3$  and  $1 \leq m \leq n - f - 2$ .

*Bulyan* first selects  $n - 2f - 2$  gradients by iterating  $n - 2f - 2$  times over *Krum*, each time removing the highest scoring gradient from the input gradient set. From these  $n - 2f - 2 \geq 2f + 1$  selected gradients, *Bulyan* outputs

<sup>3</sup>The original paper called the GAR *Multi-Krum* when  $m > 1$ .

the coordinate-wise average of the  $n - 4f - 2 \geq 1$  closest coordinate values to the coordinate-wise median.

The theoretical requirements for the  $(\alpha, f)$ -Byzantine resilience of *Bulyan* are the same as the ones of *Krum*.

### 2.3. Studied Attacks

The two, state-of-the-art attacks studied in this paper follow the same core algorithm, that we identify below.

Let  $\varepsilon_t \in \mathbb{R}_{\geq 0}$  be a non-negative factor, and  $a_t \in \mathbb{R}^d$  an *attack vector* which value depends on the actual attack used.

At each step  $t$ , each of the  $f$  Byzantine workers submits the same Byzantine gradient:  $\bar{g}_t + \varepsilon_t \cdot a_t$  (5), where  $\bar{g}_t$  is an approximation of the real gradient  $\nabla Q(\theta_t)$  at step  $t$ .

For both of the studied attacks, the value of  $\varepsilon_t$  is fixed.

#### 2.3.1. A LITTLE IS ENOUGH (BARUCH ET AL., 2019)

In this attack, a Byzantine worker submits  $\bar{g}_t + \varepsilon_t \cdot a_t$ , with  $a_t \triangleq -\sigma_t$  the opposite of the coordinate-wise standard deviation of the honest gradient distribution  $\mathcal{G}_t$ .

#### 2.3.2. FALL OF EMPIRES (XIE ET AL., 2019)

A Byzantine worker submits  $(1 - \varepsilon_t) \bar{g}_t$ , i.e.,  $a_t \triangleq -\bar{g}_t$ .

## 3. Momentum at the Workers

The Byzantine-resilience of *Krum*, *Median* and *Bulyan* rely on the honest gradients being sufficiently *clumped*. For the GARs we study, this is formalized in equations (3) and (4).

This requirement is theoretically important. When the variance of the honest gradients is *too high* compared to their norms (e.g., Equation (3) unsatisfied), the Byzantine gradients can induce aggregated gradients having negative dot-products with the real gradient, preventing convergence (as such aggregated gradients would locally increase the loss).

This requirement is also not satisfied in practice. When reproducing the attacks (figures 2 to 5), we measured and observed that the honest gradients' variance is often at least one order of magnitude too large for all the studied GARs. In the 400 experiments we performed under attack, we measured the theoretical requirement of Equation (3) is never satisfied, not even for a single step, in 394 of them. Among the 6 other experiments (all with the CIFAR-10 model), equations (3) or (4) were never verified for more than 4 steps (out of 3000) per experiment.

Nevertheless, our empirical evidences show that *reducing* the honest gradients' variance relative to their norm can be enough to defend the training against the two presented attacks. In this section we present a technique aiming at decreasing the *variance-norm ratio* of the honest gradients,

reducing or even cancelling at negligible computational costs the effects of the attacks studied in this paper.

### 3.1. Formulation

From the formulation of *momentum SGD* (Equation (2)):

$$G_t \triangleq \sum_{u=0}^t \mu^{t-u} F(g_u^{(1)}, \dots, g_u^{(n)})$$

we instead confer the momentum operation on the workers:

$$G_t \triangleq F\left(\underbrace{\sum_{u=0}^t \mu^{t-u} g_u^{(1)}}_{G_t^{(1)}}, \dots, \underbrace{\sum_{u=0}^t \mu^{t-u} g_u^{(n)}}_{G_t^{(n)}}\right) \quad (6)$$

**Notations.** In the remaining of this paper, we call the original formulation (*momentum*) *at the server*, and the proposed, revised formulation (*momentum*) *at the workers*.

### 3.2. Effects

We compare the variance-norm ratio when momentum is computed *at the server* versus *at the workers*.

Let  $\lambda_t \triangleq \|\mathbb{E} \mathcal{G}_t\| > 0$  be the real gradient's norm at step  $t$ .

Let  $\sigma_t \triangleq \sqrt{\mathbb{E} \|\mathcal{G}_t - \mathbb{E} \mathcal{G}_t\|^2}$  be the standard deviation of the real gradient at step  $t$ . The variance-norm ratio, when momentum is computed *at the server*, is:

$$r_t^{(s)} \triangleq \frac{\sigma_t^2}{\lambda_t^2}$$

We will now compute this ratio when momentum is applied at the workers. Let  $G_t^{(i)}$ , with  $G_{-1}^{(i)} \triangleq 0$ , be the gradient sent by any honest worker  $i$  at step  $t$ , i.e.:

$$G_t^{(i)} \triangleq \sum_{u=0}^t \mu^{t-u} g_u^{(i)}$$

Then, for any two honest worker identifiers  $i \neq j$ :

$$\begin{aligned} & \mathbb{E} \left\| G_t^{(i)} - G_t^{(j)} \right\|^2 \\ &= \mathbb{E} \left\| g_t^{(i)} + \mu G_{t-1}^{(i)} - g_t^{(j)} - \mu G_{t-1}^{(j)} \right\|^2 \\ &= \mathbb{E} \left\| g_t^{(i)} - g_t^{(j)} \right\|^2 + \mu^2 \mathbb{E} \left\| G_{t-1}^{(i)} - G_{t-1}^{(j)} \right\|^2 \\ & \quad + 2 \underbrace{\mu \left( \underbrace{\mathbb{E} g_t^{(i)} - \mathbb{E} g_t^{(j)}}_{=\mathbb{E} \mathcal{G}_t - \mathbb{E} \mathcal{G}_t} \right)}_{=0} \cdot \left( \mathbb{E} G_{t-1}^{(i)} - \mathbb{E} G_{t-1}^{(j)} \right) \\ &= \mathbb{E} \left\| g_t^{(i)} - g_t^{(j)} \right\|^2 + \mu^2 \mathbb{E} \left\| G_{t-1}^{(i)} - G_{t-1}^{(j)} \right\|^2 \\ &= 2 \sigma_t^2 + \mu^2 (2 \sigma_{t-1}^2 + \mu^2 (2 \sigma_{t-2}^2 + \mu^2 (\dots))) \end{aligned}$$

$$= 2 \sum_{u=0}^t \mu^{2(t-u)} \sigma_u^2 \quad (7)$$

$$= 2 \mathbb{E} \left\| G_t^{(i)} - \mathbb{E} G_t^{(i)} \right\|^2$$

$$\begin{aligned} \left\| \mathbb{E} G_t^{(i)} \right\|^2 &= \left\| \mathbb{E} g_t^{(i)} + \mu \mathbb{E} G_{t-1}^{(i)} \right\|^2 \\ &= \left\| \mathbb{E} g_t^{(i)} \right\|^2 + 2\mu \mathbb{E} g_t^{(i)} \cdot \mathbb{E} G_{t-1}^{(i)} + \mu^2 \left\| \mathbb{E} G_{t-1}^{(i)} \right\|^2 \\ &= \lambda_t^2 + 2\mu \mathbb{E} g_t^{(i)} \cdot \left( \mathbb{E} g_{t-1}^{(i)} + \mu \left( \mathbb{E} g_{t-2}^{(i)} + \mu (\dots) \right) \right) \\ &\quad + \mu^2 \left( \lambda_{t-1}^2 + 2\mu \mathbb{E} g_{t-1}^{(i)} \cdot \left( \mathbb{E} g_{t-2}^{(i)} + \mu (\dots) \right) \right. \\ &\quad \left. + \mu^2 \mathbb{E} \left\| G_{t-2}^{(i)} \right\|^2 \right) \\ &= \sum_{u=0}^t \mu^{2(t-u)} \left( \lambda_u^2 + 2 \sum_{v=0}^{u-1} \mu^{u-v} \underbrace{\mathbb{E} g_u^{(i)} \cdot \mathbb{E} g_v^{(i)}}_{=\mathbb{E} \mathcal{G}_u \cdot \mathbb{E} \mathcal{G}_v} \right) \end{aligned}$$

Thus, assuming honest gradients  $\mathbb{E} G_t^{(i)}$  do not become null:

$$r_t^{(w)} \triangleq \frac{\Omega_t^2}{\Lambda_t^2} = \frac{\sum_{u=0}^t \mu^{2(t-u)} \sigma_u^2}{\sum_{u=0}^t \mu^{2(t-u)} (\lambda_u^2 + s_u)}$$

where the expected ‘‘straightness’’ of the gradient computed by an honest worker at step  $u$  is defined by:

$$s_u \triangleq 2 \sum_{v=0}^{u-1} \mu^{u-v} \mathbb{E} \mathcal{G}_u \cdot \mathbb{E} \mathcal{G}_v$$

$s_u$  quantifies what can be thought as the *curvature* of the honest gradient trajectory. Straight trajectories can make  $s_u$  grow up to  $(1 - \mu)^{-1} > 1$  times the expected squared-norm of the honest gradients, while highly ‘‘curved’’ trajectories (e.g., close to a local minimum) tend to make  $s_u$  negative.

This observation stresses that this formulation of momentum can sometimes be harmful for the purpose of Byzantine resilience. We measured  $s_u$  for every step  $u > 0$  in our experiments, and we always observed that this quantity is positive and increases for a short window of (dozen) steps (depending on  $\eta_t$ ), and then oscillates between positive and negative values. These two phases are noticeable in the first steps of Figure 5. While the empirical impact (decreased or cancelled loss in accuracy) is concrete, we believe there is room for further improvements, discussed in Section 5.

The purpose of using momentum at the workers is to reduce the variance-norm ratio  $r_t^{(w)}$ , compared to  $r_t^{(s)}$ . Since  $g_0^{(i)} = G_0^{(i)}$ , we verify that  $r_0^{(u)} = r_0^{(w)}$ . Then,  $\forall t > 0$ :

$$\begin{aligned} r_t^{(w)} \leq r_t^{(s)} &\Leftrightarrow \frac{\sigma_t^2 + \mu^2 \Omega_{t-1}^2}{\lambda_t^2 + s_t + \mu^2 \Lambda_{t-1}^2} \leq \frac{\sigma_t^2}{\lambda_t^2} \\ &\Leftrightarrow \mu^2 \Omega_{t-1}^2 \lambda_t^2 \leq (s_t + \mu^2 \Lambda_{t-1}^2) \sigma_t^2 \end{aligned}$$

$$\Leftrightarrow s_t \geq \mu^2 \Lambda_{t-1}^2 \left( \frac{r_{t-1}^{(w)}}{r_t^{(s)}} - 1 \right) \quad (8)$$

The condition for decreasing  $r_t^{(w)}$  can be obtained similarly:

$$r_t^{(w)} \leq r_{t-1}^{(w)} \Leftrightarrow s_t \geq \lambda_t^2 \left( \frac{r_t^{(s)}}{r_{t-1}^{(s)}} - 1 \right)$$

To study the impact of a lower learning rate  $\eta_t$  on  $s_t$ , we will assume that the real gradient  $\nabla Q$  is  $l$ -Lipschitz. Namely:

$$\begin{aligned} \forall (t, u) \in \mathbb{N}^2, u < t, \|\mathbb{E} \mathcal{G}_t - \mathbb{E} \mathcal{G}_u\|^2 &\leq l^2 \|\theta_t - \theta_u\| \\ &\leq l^2 \left\| \sum_{v=u}^{t-1} \eta_v G_v \right\| \end{aligned}$$

Then,  $\forall (t, u) \in \mathbb{N}^2, u < t$ , we can rewrite:

$$\|\mathbb{E} \mathcal{G}_t - \mathbb{E} \mathcal{G}_u\|^2 = \underbrace{\|\mathbb{E} \mathcal{G}_t\|^2}_{\lambda_t^2} + \underbrace{\|\mathbb{E} \mathcal{G}_u\|^2}_{\lambda_u^2} - 2 \mathbb{E} \mathcal{G}_t \cdot \mathbb{E} \mathcal{G}_u$$

And finally, we can lower-bound  $s_t$  as:

$$\begin{aligned} &\sum_{u=0}^{t-1} \mu^{t-u} \|\mathbb{E} \mathcal{G}_t - \mathbb{E} \mathcal{G}_u\|^2 \\ &= \sum_{u=0}^{t-1} \mu^{t-u} (\lambda_t^2 + \lambda_u^2) - 2 \underbrace{\sum_{u=0}^{t-1} \mu^{t-u} \mathbb{E} \mathcal{G}_t \cdot \mathbb{E} \mathcal{G}_u}_{s_t} \\ &\leq \sum_{u=0}^{t-1} \mu^{t-u} l^2 \left\| \sum_{v=u}^{t-1} \eta_v G_v \right\| \\ \Leftrightarrow s_t &\geq \sum_{u=0}^{t-1} \mu^{t-u} \left( \lambda_t^2 + \lambda_u^2 - l^2 \left\| \sum_{v=u}^{t-1} \eta_v G_v \right\| \right) \quad (9) \\ &\geq \frac{1 - \mu^t}{1 - \mu} \lambda_t^2 + \sum_{u=0}^{t-1} \mu^{t-u} \left( \lambda_u^2 - l^2 \left\| \sum_{v=u}^{t-1} \eta_v G_v \right\| \right) \end{aligned}$$

When the real gradient  $\nabla Q$  is (locally) Lipschitz continuous, reducing the learning rate  $\eta_t$  can suffice to ensure  $s_t$  satisfies the conditions laid above for decreasing the variance-norm ratio  $r_t^{(w)}$ ; the purpose of momentum at the workers.

Importantly this last lower bound, namely Equation (9), sets how the practitioner should choose two hyperparameters,  $\mu$  and  $\eta_t$ , for the purpose of Byzantine-resilience. Basically, and as long as it does not harm the training without adversary,  $\mu$  should be set as *high* and  $\eta_t$  as *low* as possible.

As a side note, (Xie et al., 2019) is prone to increasing the lower bound on  $s_t$ . Indeed, this attack submits gradients smaller or opposed to the honest gradient (Section 2.3.2). Such an attack can shorten the parameter trajectory, and so can improve Byzantine-resilience in the ensuing step(s).

## 4. Experiments

The goal of this section is to empirically verify our theoretical results, measuring the evolution of both the top-1 cross-accuracy and variance-norm ratio over the training. Our experiments cover every possible combinations of 5 key hyperparameters, including combinations used by (Baruch et al., 2019; Xie et al., 2019). For reproducibility and confidence in the results, each combination of hyperparameters is repeated 5 times with *seeds* 1 to 5, totalling 440 different runs. Besides observing the benefit of lower learning rates, our results show tangible mitigation of both attacks.

### 4.1. Experimental Setup

We use a compact notation to define the models: L(#outputs) for a fully-connected linear layer, R for ReLU activation, S for log-softmax, C(#channels) for a fully-connected 2D-convolutional layer (kernel size 3, padding 1, stride 1), M for 2D-maxpool (kernel size 2), N for batch-normalization, and D for dropout (with fixed probability 0.25).

We use the model and dataset from (Baruch et al., 2019):

**Model** (784)-L(100)-R-L(10)-R-S  
**Dataset** MNIST (83 training points/gradient)  
**#workers**  $n = 51$   $f \in \{24, 12\}$

We also use the model and dataset from (Xie et al., 2019):

**Model** (3,  $32 \times 32$ )-C(64)-R-B-C(64)-R-B-M-D-  
 -C(128)-R-B-C(128)-R-B-M-D-  
 -L(128)-R-D-L(10)-S  
**Dataset** CIFAR-10 (50 training points/gradient)  
**#workers**  $n = 25$   $f \in \{11, 5\}$

For model training, we use the *negative log likelihood* loss and respectively  $10^{-4}$  and  $10^{-2}$   $\ell_2$ -regularization for the MNIST and CIFAR-10 models. We also clip gradients, ensuring their norms remain respectively below 2 and 5 for the MNIST and CIFAR-10 models. For model evaluation, we use the *top-1 cross-accuracy* on the whole testing set.

Both datasets are pre-processed before training. For MNIST we apply the same pre-processing as in (Baruch et al., 2019): an input image normalization with mean 0.1307 and standard deviation 0.3081. For CIFAR-10, besides including horizontal flips of the input pictures, we also apply a per-channel normalization with means 0.4914, 0.4822, 0.4465 and standard deviations 0.2023, 0.1994, 0.2010 (Liu, 2019).

We set  $f$  the number of Byzantine workers either to the maximum for which *Krum* can be used (roughly an half:  $f = \lfloor \frac{n-3}{2} \rfloor$ ), or the maximum for *Bulyan* (roughly a quarter,  $f = \lfloor \frac{n-3}{4} \rfloor$ ). The attack factors  $\varepsilon_t$  (Section 2.3) are set to constants proposed in the literature, namely  $\varepsilon_t = 1.5$  for (Baruch et al., 2019) and  $\varepsilon_t = 1.1$  for (Xie et al., 2019).

Guided by our theoretical study on the impact of the learning

rate on the variance-norm ratio, for every pair model-attack in our experiments we select two different learning rates. The first and largest is selected so as to maximize the *performance* (highest final cross-accuracy and accuracy gain per step) of the model trained without Byzantine workers. The second and smallest is chosen so as to minimize the *performance loss* under attack, without substantially impacting the final accuracy when trained without Byzantine workers.

The MNIST and CIFAR-10 model are trained respectively with  $\mu = 0.9$  and  $\mu = 0.99$ . These values were obtained by trial and error, to maximize overall accuracy gain per step.

Our theoretical analysis highlights two metrics: the *top-1 cross-accuracy*, measuring the performance of the model, and the *variance-norm ratio*, i.e. either  $r_t^{(s)}$  or  $r_t^{(w)}$  in accordance with where momentum was carried out. Each experiment is run 5 times. We present the average and standard deviation of the two metrics over these 5 runs.

### 4.2. Reproducibility

Particular care has been taken to make our results reproducible. Each of the 5 runs per experiment are respectively seeded with seed 1 to 5. For instance, this implies that two experiments with same seed and same model also starts with the same parameters  $\theta_0$ . To further reduce the sources of non-determinism, the CuDNN backend is configured in deterministic mode (our experiments ran on a *GeForce GTX 1080 Ti*) with benchmark mode turned off. We also used *log-softmax + nll loss*, which is equal to *softmax + cross-entropy loss*, but with improved numerical stability on PyTorch.

We provide our code along with a script reproducing *all* of our results, both the experiments and the graphs, in one command. Details, including software and hardware dependencies, are available in the supplementary material.

### 4.3. Experimental Results

For each of the pair model-dataset, we consider 5 variable hyperparameters: which attack to test ((Baruch et al., 2019) or (Xie et al., 2019)), which defense to run (*Krum*, *Median* or *Bulyan*), how many Byzantine workers  $f$  to use (*an half* or *a quarter*), where momentum is computed (*at the server* or *at the workers*) and which learning rate  $\eta_t$  to apply.

We report on every possible combination of these hyperparameters, along with baselines that use *averaging* without attack. With 5 repetitions per setup, the experiments consist in 440 runs, aggregated and studied in the supplementary material. In this section we report on a representative subset.

We made a concerning observation: one of the theoretical requirement for Byzantine-resilience, equations (3) or (4), is actually rarely satisfied in practice. In less than 2% of the runs under attack was this theoretical condition satisfied for

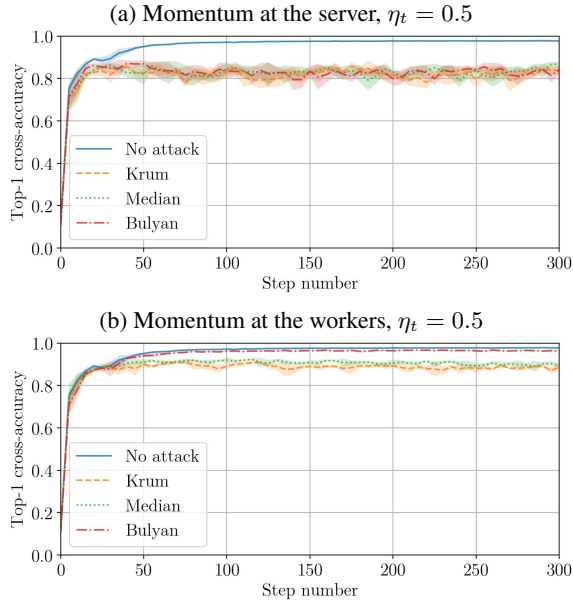


Figure 2. MNIST and the associated model (Section 4.1),  $n = 51$  and  $f = 12$ . The attack, (Baruch et al., 2019), has a tangible impact ( $-15\%$  on the maximum accuracy) in this setup. Using momentum at the workers diminishes the effect of the attack, even substantially when *Bulyan* is used (only  $1\%$  loss in accuracy).

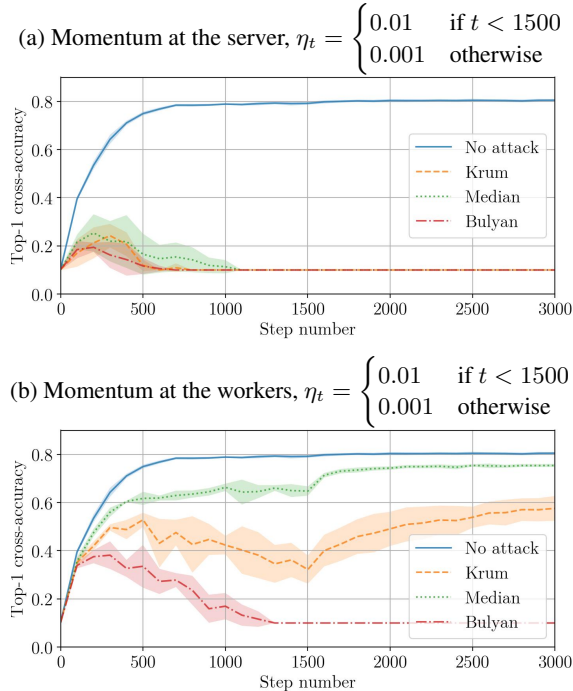


Figure 3. CIFAR-10 and the associated model (Section 4.1), with  $n = 25$  and  $f = 5$ . As in Figure 2, (Baruch et al., 2019) has a strong impact on the training. The positive effect of momentum at the workers (Figure 3b) is conspicuous and, as predicted by the theory (Section 3.2), amplified with a lower learning rate.

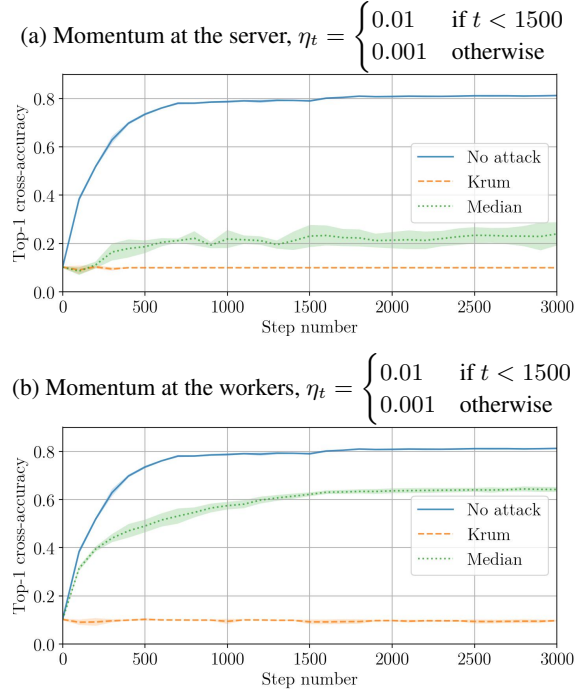


Figure 4. CIFAR-10 and the associated model (Section 4.1), with  $n = 25$  and  $f = 11$  (Figure 3 uses  $f = 5$ ). The attack, (Xie et al., 2019), is extremely efficient when  $f \approx \frac{n}{2}$ , but had almost no effect with  $f \approx \frac{n}{4}$ ; the supplementary material has a more complete range of experiments. Momentum at the workers noticeably improves the cross-accuracy when *Median* is used in this setting.

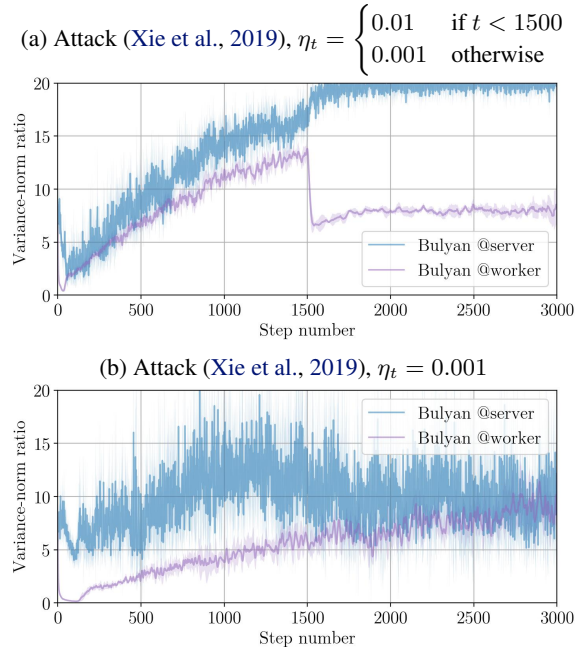


Figure 5. CIFAR-10 and the associated model (Section 4.1), with  $n = 25$  and  $f = 5$ , showing a decreased variance-norm ratio when momentum is computed at the workers. The benefit of a lower  $\eta_t$  is clearly visible in (a), when its value is reduced at step 1500.

at least 1 step, and none for more than 4 steps (out of 3000). In most of the experiments, the observed variance-norm ratio was often between 1 and 2 *orders of magnitude* too high (e.g., Figure 5). Since our hyperparameters (model, dataset, mini-batch size,  $n$ ,  $f$ ) are very close, if not equal, to those used in the experiments of (Baruch et al., 2019; Xie et al., 2019), such a substantial margin (1 to 2 orders of magnitude) lets us think the theoretical requirements for Byzantine resilience (equations (3) or (4)) were actually not satisfied either in (Baruch et al., 2019; Xie et al., 2019).

**Result Analysis.** With the largest, optimal learning rate, we obtained with *Krum* and for both models very similar maximum top-1 cross-accuracies to the ones obtained in (Baruch et al., 2019)<sup>4</sup>, namely  $\sim 80\%$  for MNIST’s model (Figure 2a) and  $\sim 20\%$  for CIFAR-10’s model (Figure 3a).

A similar observation can be made for (Xie et al., 2019): on the CIFAR-10’s model, the maximum accuracies of *Krum* and *Median* are respectively  $\sim 10\%$  and  $\sim 20\%$  (Figure 4a).

The benefit of computing momentum at the workers is visible in all the figures. Regarding the impact on the top-1 cross-accuracy, we systematically<sup>5</sup> observe an increase compared to when momentum is computed at the server (figures 2–4). The empirical increase ranges from  $+5\%$ , on the MNIST model attacked by (Baruch et al., 2019) (supplementary material, Figure 3), to  $+50\%$  on the CIFAR-10 model, defended by *Median* (Figure 3). Regarding the effect on the variance-norm ratio, we comparatively observe a decrease of this ratio before approaching convergence. As predicted by our theoretical analysis, this decrease can be amplified by reducing the learning rate (Figure 5a).

## 5. Concluding Remarks

**Momentum-based Variance Reduction.** Our algorithm is different from (Cutkosky & Orabona, 2019), as instead of reducing the variance of the gradients, we actually *increase* it (Equation (7)). What we seek to reduce is the *variance-norm ratio*, which is the key quantity for any Byzantine-resilient GAR approximating a high-dimensional median, e.g. *Krum*, *Median*, *Bulyan* as well as in (Yang & Bajwa, 2019b;a; Chen et al., 2017; Muñoz-González et al., 2019)<sup>6</sup>.

Some of the ideas introduced in (Cutkosky & Orabona, 2019) could nevertheless help further improve Byzantine resilience. For instance, introducing an adaptive learning rate which decreases depending on the curvature of the parameter trajectory is an appealing approach to further reduce the variance-norm ratio (Equation (9)).

<sup>4</sup>Although (Baruch et al., 2019) uses *Krum* with  $m = 1$  (see Section 2.2.1), and not the exact same CIFAR-10 model.

<sup>5</sup>Except for *Krum* against (Xie et al., 2019) when  $f = \lfloor \frac{n-3}{2} \rfloor$ .

<sup>6</sup>This list is not exhaustive.

**Further Work.** The theoretical condition for ratio reduction, in Section 3.2, shows that momentum at the workers is a double-edged sword. The intuition can be gained with the classic analogy from physics: without Byzantine workers, momentum makes the parameters  $\theta_t$  somehow like a particle travelling down the loss function with *inertia*. When inside a “straight valley”, past estimation errors are on average compensated in the next steps, dampening oscillations and accumulating the average descent direction. The variance-norm ratio of the momentum gradient is then reduced, mostly because its norm *increases*, which is quantified by  $s_t$ . The problem is that  $s_t$  can become negative. Intuitively with the particle analogy, this happens when the loss is locally “curved”, for instance when approaching a local minimum. The particle may start continuing “uphill” instead of following the “valley”, and so, losing *momentum*. The norm of the momentum gradient then *decreases*, increasing the variance-norm ratio.

While the ability to cross narrow, local minima is recognized as an accelerator (Goh, 2017), for the purpose of Byzantine-resilience we want to ensure momentum at the workers does not increase the variance-norm ratio compared to the classical, momentum at the server. The theoretical condition for this purpose is given in Equation (8). One simple amendment would then be to use momentum at the workers when Equation (8) is satisfied, and fallback to computing it at the server otherwise. Also, a more complex, possible future approach could be to dynamically adapt the momentum factor  $\mu$ , decreasing it as the curvature increases.

**Asynchronous SGD.** We focused in this work on the synchronous setting, which received most of the attention in the Byzantine-resilient literature. Yet, we believe our work can be applied to asynchronous settings, as momentum is agnostic to the question of synchrony. Specifically, combining our idea with a filtering scheme such as *Kardam* (Damaskinos et al., 2018) is in principle possible, as the filter commutes with the basic operations of momentum. However, further analysis of the interplay between the dynamics of stale gradients and the dynamics of momentum remain necessary.

**Byzantine Servers.** While most of the research on Byzantine-resilience gradient descent has focused on the workers’ side, assuming a reliable server, recent efforts have started tackling Byzantine servers (El-Mhamdi et al., 2019). Our reduction of the variance-norm ratio strengthens the gradient aggregation phase, which is necessary whether we deal with Byzantine workers or Byzantine servers. An interesting open question is how the momentum dynamics affects the models drift between different parameter servers. Any quantitative answer to this question will enable the use of our method in fully decentralised Byzantine resilient gradient descent.



## References

- Alistarh, D., Allen-Zhu, Z., and Li, J. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 4618–4628, 2018.
- Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. How to backdoor federated learning. *CoRR*, abs/1807.00459, 2018.
- Baruch, M., Baruch, G., and Goldberg, Y. A little is enough: Circumventing defenses for distributed learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, 8-14 December 2019, Long Beach, CA, USA, 2019*.
- Bernstein, J., Zhao, J., Azizzadenesheli, K., and Anandkumar, A. signsgd with majority vote is communication efficient and fault tolerant. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, 2019*.
- Blanchard, P., El-Mhamdi, E.-M., Guerraoui, R., and Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 119–129, 2017.
- Bottou, L. Online learning and stochastic approximations. *Online learning in neural networks*, 17(9):142, 1998.
- Chen, L., Wang, H., Charles, Z. B., and Papailiopoulos, D. S. DRACO: byzantine-resilient distributed training via redundant gradients. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 902–911, 2018.
- Chen, Y., Su, L., and Xu, J. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *CoRR*, abs/1705.05491, 2017.
- Cutkosky, A. and Orabona, F. Momentum-based variance reduction in non-convex SGD. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d’Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pp. 15210–15219, 2019. URL <http://papers.nips.cc/paper/9659-momentum-based-variance-reduction-in-non-convex-sgd>.
- Damaskinos, G., El-Mhamdi, E.-M., Guerraoui, R., Patra, R., and Taziki, M. Asynchronous byzantine machine learning (the case of SGD). In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 1153–1162, 2018.
- Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., et al. Large scale distributed deep networks. In *NIPS*, pp. 1223–1231, 2012.
- El-Mhamdi, E.-M., Guerraoui, R., and Rouault, S. The hidden vulnerability of distributed learning in byzantium. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 3518–3527, 2018.
- El-Mhamdi, E.-M., Guerraoui, R., Guirguis, A., and Rouault, S. Sgd: Decentralized Byzantine resilience. *arXiv preprint arXiv:1905.03853*, 2019.
- Goh, G. Why momentum really works. *Distill*, 2017. doi: 10.23915/distill.00006. URL <http://distill.pub/2017/momentum>.
- Konečný, J., McMahan, B., and Ramage, D. Federated optimization: Distributed optimization beyond the datacenter. *CoRR*, abs/1511.03575, 2015.
- Lamport, L., Shostak, R. E., and Pease, M. C. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982. doi: 10.1145/357172.357176.
- Li, M., Andersen, D. G., Park, J. W., Smola, A. J., Ahmed, A., Josifovski, V., Long, J., Shekita, E. J., and Su, B. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on Operating Systems Design and Implementation, OSDI ’14, Broomfield, CO, USA, October 6-8, 2014*, pp. 583–598, 2014.
- Lian, X., Huang, Y., Li, Y., and Liu, J. Asynchronous parallel stochastic gradient for nonconvex optimization. In *NIPS*, pp. 2737–2745, 2015.
- Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, B. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=SkhQHMW0W>.
- Liu, K. Train cifar-10 with pytorch, 2019. URL <https://github.com/kuangliu/pytorch-cifar/blob/ab908327d44bf9b1d22cd333a4466e85083d3f21/main.py#L33>.

- Muñoz-González, L., Co, K. T., and Lupu, E. C. Byzantine-robust federated machine learning through adaptive model averaging. *arXiv preprint arXiv:1909.05125*, 2019.
- Rajput, S., Wang, H., Charles, Z., and Papailiopoulos, D. Detox: A redundancy-based framework for faster and more robust gradient aggregation. *Neural Information Processing Systems*, 2019.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct 1986. doi: 10.1038/323533a0.
- Schneider, F. B. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*, 22(4):299–319, 1990.
- Sun, Z., Kairouz, P., Suresh, A. T., and McMahan, H. B. Can you really backdoor federated learning? *CoRR*, abs/1911.07963, 2019.
- TianXiang, W., ZHENG, Z., ChangBing, T., and Hao, P. Aggregation rules based on stochastic gradient descent in byzantine consensus. In *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, pp. 317–324. IEEE, 2019.
- Xie, C., Koyejo, O., and Gupta, I. Generalized Byzantine-tolerant sgd. *arXiv preprint arXiv:1802.10116*, 2018a.
- Xie, C., Koyejo, O., and Gupta, I. Phocas: dimensional byzantine-resilient stochastic gradient descent. *arXiv preprint arXiv:1805.09682*, 2018b.
- Xie, C., Koyejo, O., and Gupta, I. Fall of empires: Breaking byzantine-tolerant SGD by inner product manipulation. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*, pp. 83, 2019.
- Xie, C., Huang, K., Chen, P.-Y., and Li, B. {DBA}: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=rkgyS0VFvr>.
- Yang, Z. and Bajwa, W. U. Bridge: Byzantine-resilient decentralized gradient descent. *arXiv preprint arXiv:1908.08098*, 2019a.
- Yang, Z. and Bajwa, W. U. Byrdie: Byzantine-resilient distributed coordinate descent for decentralized learning. *IEEE Transactions on Signal and Information Processing over Networks*, 2019b.
- Yang, Z., Gang, A., and Bajwa, W. U. Adversary-resilient inference and machine learning: From distributed to decentralized. *arXiv preprint arXiv:1908.08649*, 2019.
- Yin, D., Chen, Y., Ramchandran, K., and Bartlett, P. L. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, July 10-15, 2018*, pp. 5636–5645, 2018.
- Zhang, R., Zheng, S., and Kwok, J. T. Asynchronous distributed semi-stochastic gradient optimization. In *AAAI*, pp. 2323–2329, 2016.

## A. Reproducing the results

The codebase is available at <https://github.com/LPD-EPFL/ByzantineMomentum>.

### A.1. Dependencies

**Software dependencies.** Python 3.7.3 has been used to run our scripts. Besides the standard libraries associated with Python 3.7.3, our scripts also depend on:

Library	Version	Library	Version
numpy	1.17.2	six	1.12.0
torch	1.2.0	pytz	2019.3
torchvision	0.4.0	dateutil	2.7.3
pandas	0.25.1	pyparsing	2.2.0
matplotlib	3.0.2	cycler	0.10.0
tqdm	4.40.2	kiwisolver	1.0.1
PIL	6.1.0	cff	1.13.2

We list below the OS on which our scripts have been tested:

- Debian 10 (GNU/Linux 4.19.0-6 x86\_64)
- Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-58 x86\_64)

**Hardware dependencies.** Although our experiments are time-agnostic, we list below the hardware components used:

- 1 Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz
- 2 Nvidia GeForce GTX 1080 Ti
- 64 GB of RAM

### A.2. Command

Our results, i.e. the experiments and graphs, are reproducible in one command. In the root directory, please run:

```
$ python3 reproduce.py
```

On our hardware, reproducing the results takes ~24 hours.

## B. Experimental results

For every pair model-dataset, the following parameters vary:

- Which attack: (Baruch et al., 2019) or (Xie et al., 2019)
- Which defense: *Krum*, *Median* or *Bulyan*
- How many Byzantine workers (*an half* or *a quarter*)
- Where momentum is computed (*server* or *workers*)
- Which learning rate is used (*larger* or *smaller*)

Every possible combination is tested<sup>7</sup>, leading to a total of 88 different experiment setups. Each setup is tested 5 times, each run with a fixed seed from 1 to 5, enabling verbatim reproduction of our results<sup>8</sup>. We then report the average and standard deviation for two metrics: *top-1 cross-accuracy* and *variance-norm ratio* over the training steps.

The results regarding the cross-accuracy are layed out by “blocks” of 4 experiment setups presenting the same model, dataset, number of Byzantine workers and attack. These results are presented from figures 6 to 13. In each “block”, the 2 top experiments use the larger learning rate and the 2 bottom ones the smaller, so looking below correspond to looking to the same experiment but with a smaller learning rate (and vice versa). Similarly, the 2 left experiments use momentum at the server, while the 2 right ones use momentum at the workers, which allows for handy comparison of the effect of using one technique over the other.

The results regarding the variance-norm ratio are also layed out by “blocks” of 4 experiment setups presenting the same model, dataset, number of Byzantine workers and defense. These results are presented from figures 14 to 23. In each “block”, the attack from (Baruch et al., 2019) is use on the left column and (Xie et al., 2019) on the right. As for the cross-accuracy, the top row use the larger learning rate and the bottom row shows the effect of using a smaller one.

For figures 6 to 23, the captions present to the reader the hyperparameters used in each of the experiments, along with comments about the observed behaviors.

<sup>7</sup>Along with baselines using *averaging* without attack.

<sup>8</sup>Despite our best efforts, there may still exist minor sources of non-determinism, like race-conditions in the evaluation of certain functions (e.g., parallel additions) in a GPU. Nevertheless we believe these should not affect the results in any significant way.

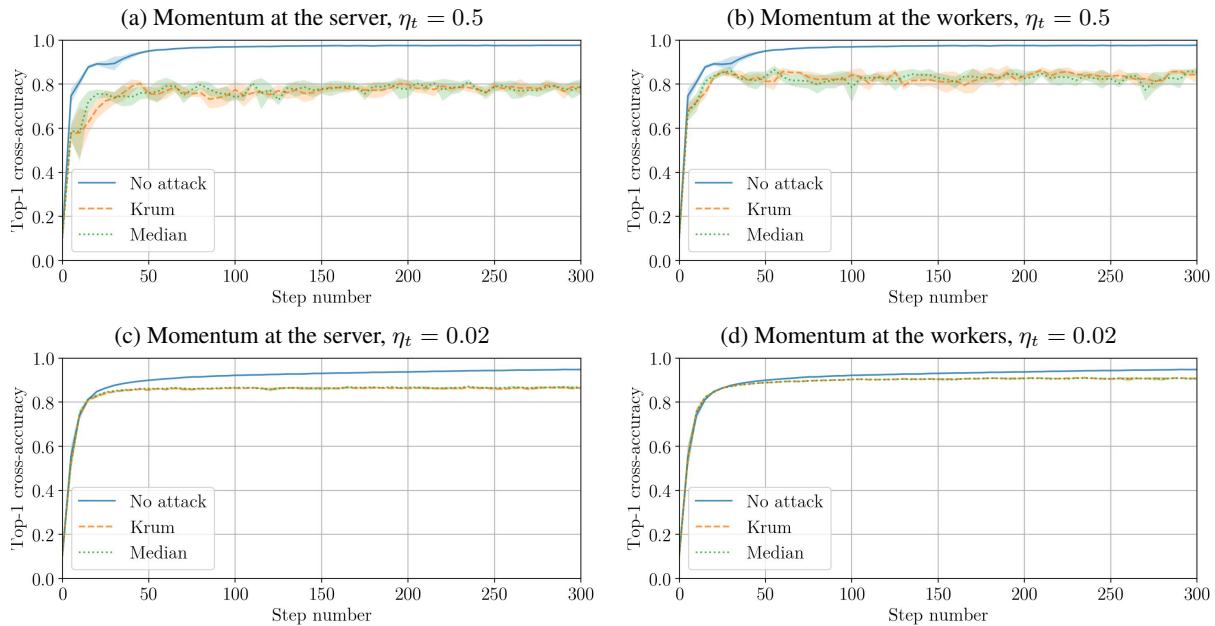


Figure 6. MNIST using  $n = 51$  workers, including  $f = 24$  Byzantine workers implementing (Baruch et al., 2019). This is the maximum number of Byzantine workers *Krum* can support. No matter where the momentum is computed, reducing the learning rate decreases the effect of the attack against all the GARs. This is not observed, in the same setting, when (Xie et al., 2019) is used instead (Figure 8). No matter the learning rate, using momentum at the workers always leads in these settings to an increase of the final accuracy (+5%).

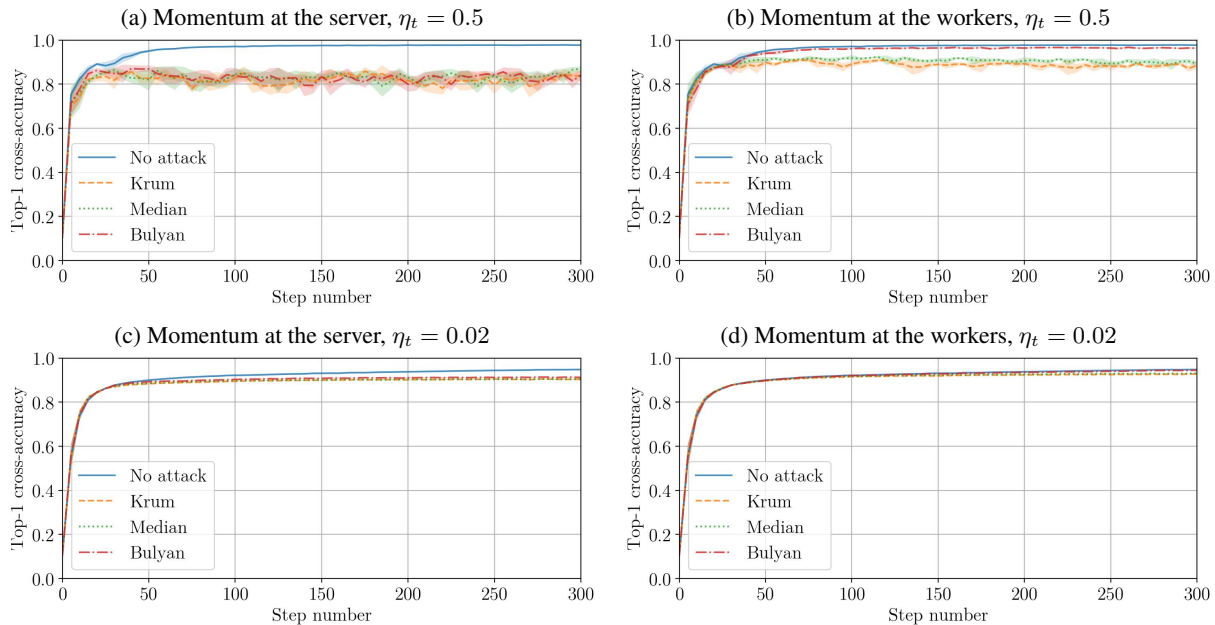


Figure 7. MNIST using  $n = 51$  workers, including  $f = 12$  Byzantine workers implementing (Baruch et al., 2019). This is the maximum number of Byzantine workers *Bulyan* can support. With momentum at the server and the largest learning rate, the impact of the attack remains unchanged compared to Figure 6 and despite the reduced number of Byzantine workers. *Bulyan*, which combines *Krum* and *Median*, achieves the same performance as both *Krum* and *Median*. When momentum is computed at the workers, *Bulyan* achieves in these settings better resilience than its parts *Krum* and *Median*, and the attack has no tangible effect anymore.

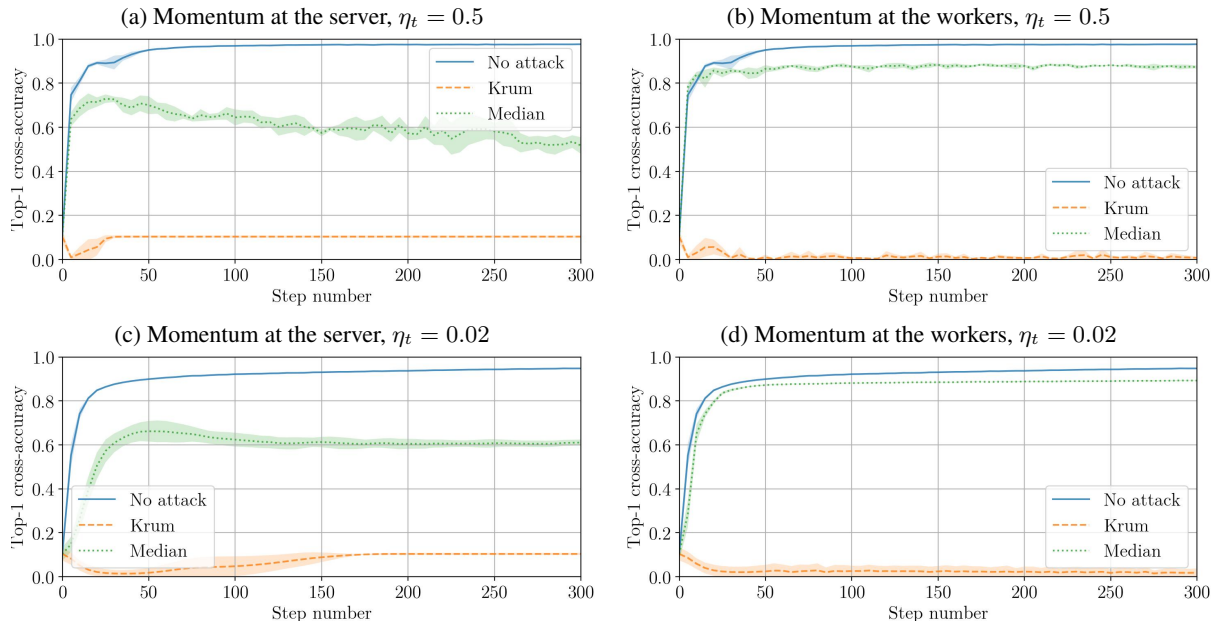


Figure 8. MNIST using  $n = 51$  workers, including  $f = 24$  Byzantine workers implementing (Xie et al., 2019). This is the maximum number of Byzantine workers *Krum* can support. Contrary to Figure 6, this attack has very different impacts on the training depending on the Byzantine-resilient GAR used. With *Krum* and momentum at the server, the model parameters quickly (after 30 steps) reach a point where the output class becomes independent from the input; the model is driven useless. Momentum at the workers with *Krum* only avoids obtaining such a model. *Median* shows substantially more resilience than *Krum* in this setup, and when momentum is computed at the workers, the maximum cross-accuracy is consistently increased, between 15% to 25% additional points.

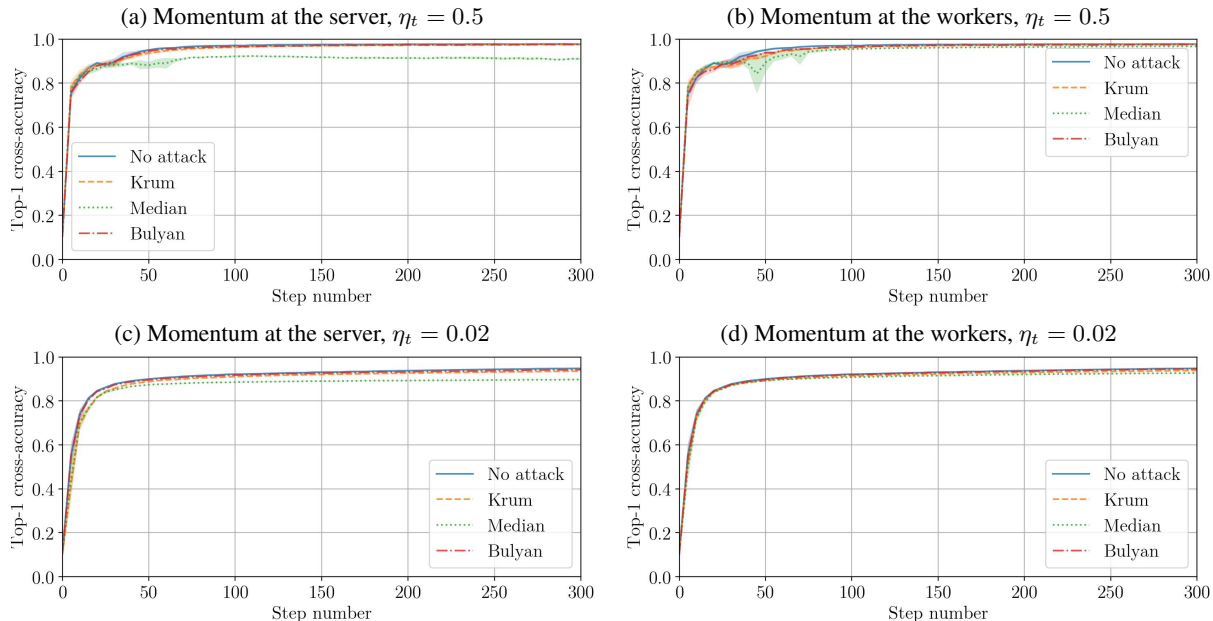


Figure 9. MNIST using  $n = 51$  workers, including  $f = 12$  Byzantine workers implementing (Xie et al., 2019). This is the maximum number of Byzantine workers *Bulyan* can support. With a quarter of Byzantine workers, the attack does not have any tangible impact on *Krum* (and thus none on *Bulyan* either) for this model and dataset anymore. Using momentum at the workers further reduces the impact on *Median*, to the point of filtering out the adversarial effect of this attack.

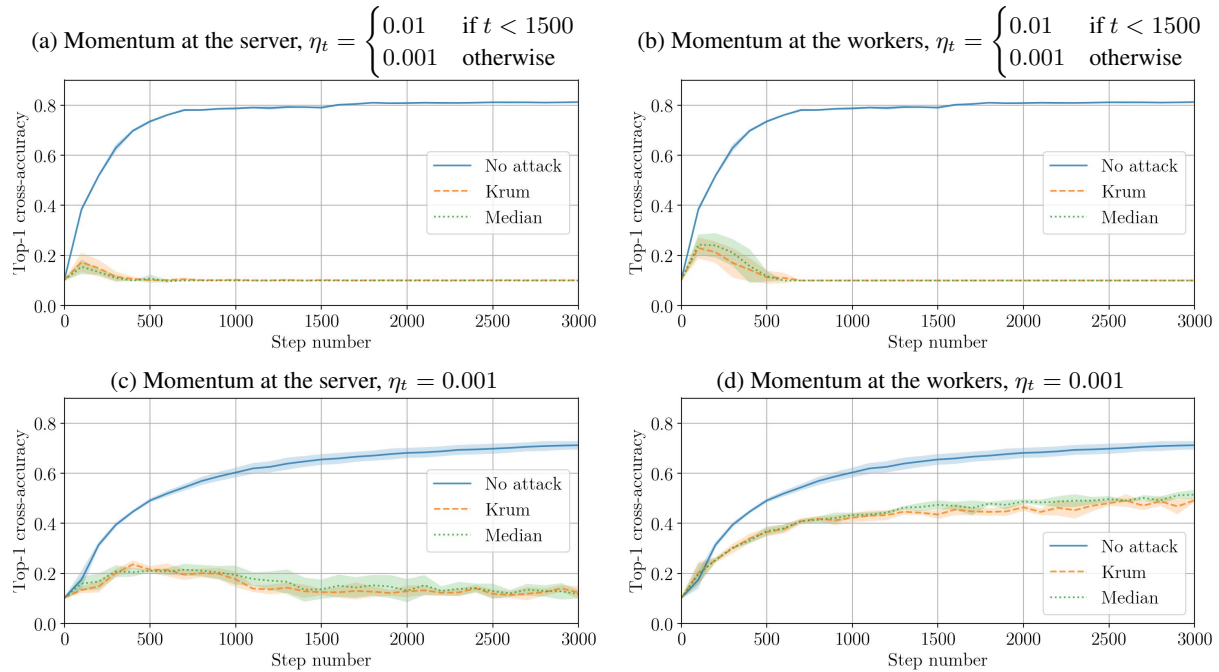


Figure 10. CIFAR-10 using  $n = 25$  workers, including  $f = 11$  Byzantine workers implementing (Baruch et al., 2019). This is the maximum number of Byzantine workers *Krum* can support. Compared to Figure 6, the impact of the attack is substantial. Even with a reduced learning rate, the model maximum cross-accuracy barely reaches 20%. Using momentum at the workers, while having virtually no computational cost, positively impacts the performance of the model.

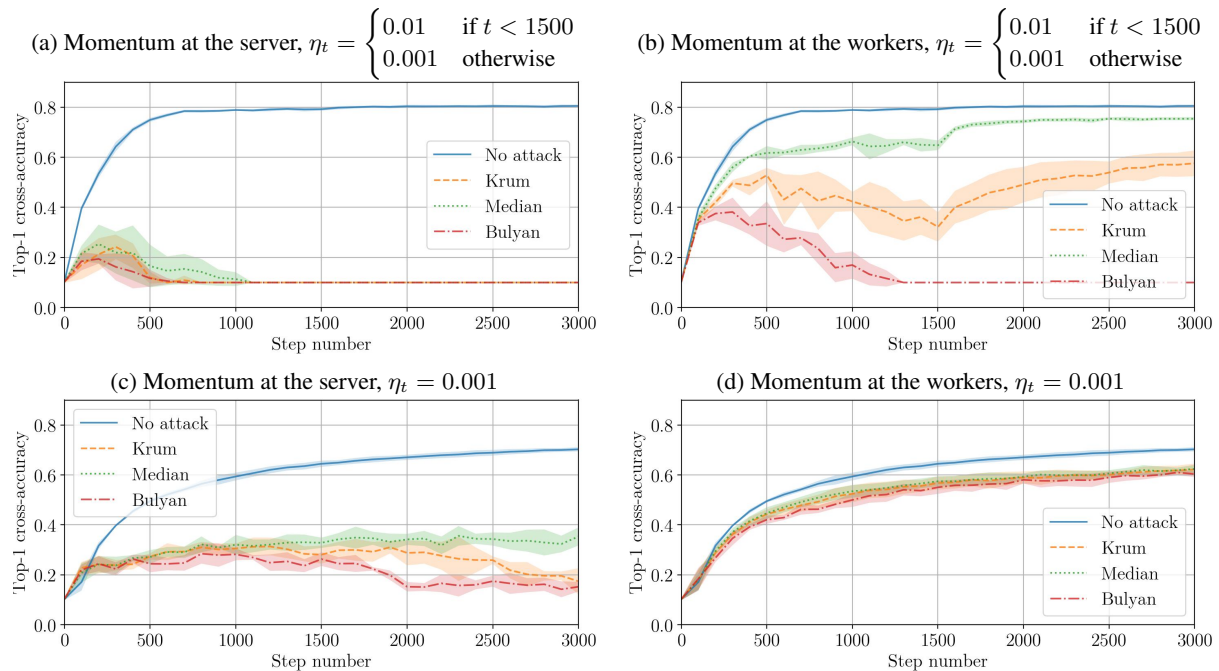


Figure 11. CIFAR-10 using  $n = 25$  workers, including  $f = 5$  Byzantine workers implementing (Baruch et al., 2019). This is the maximum number of Byzantine workers *Bulyan* can support. The effect of going to only a quarter of Byzantine workers, compared to Figure 10, did not made the attack less effective. Conversely, momentum at the workers leads to a conspicuous improvement of the model performance.

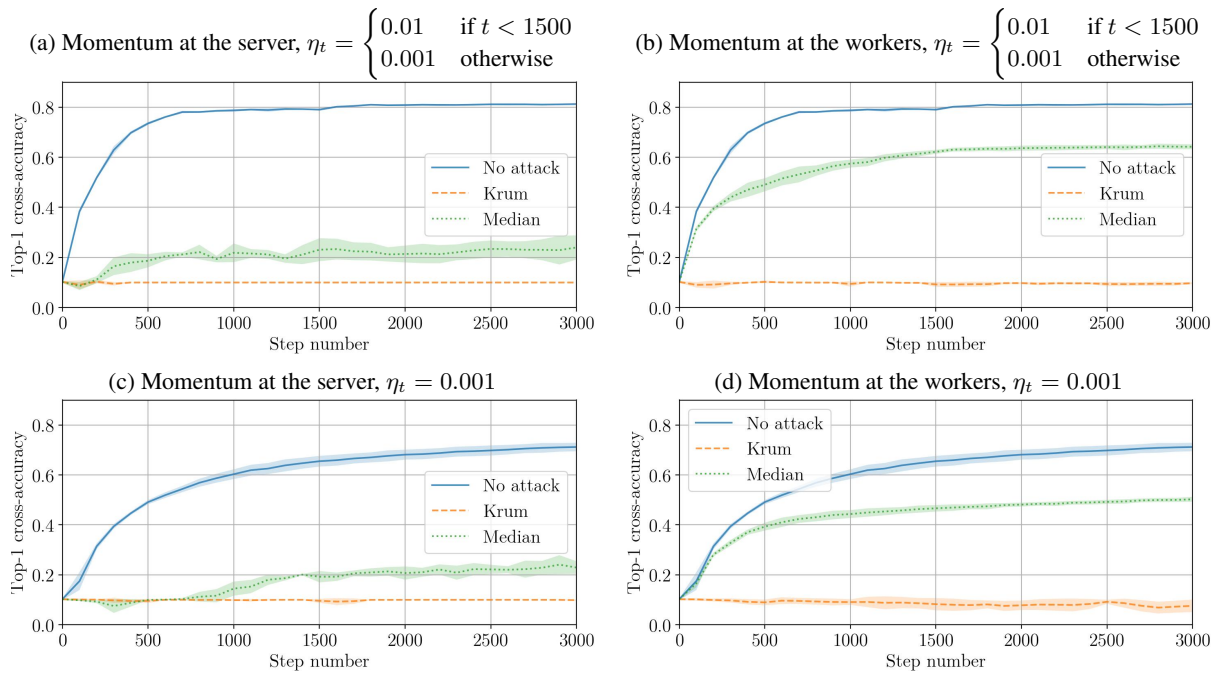


Figure 12. CIFAR-10 using  $n = 25$  workers, including  $f = 11$  Byzantine workers implementing (Xie et al., 2019). This is the maximum number of Byzantine workers *Krum* can support. As in Figure 8, the attack is extremely effective on *Krum* and slightly less on *Median*. Momentum at the workers has a substantial positive effect when *Median* is used.

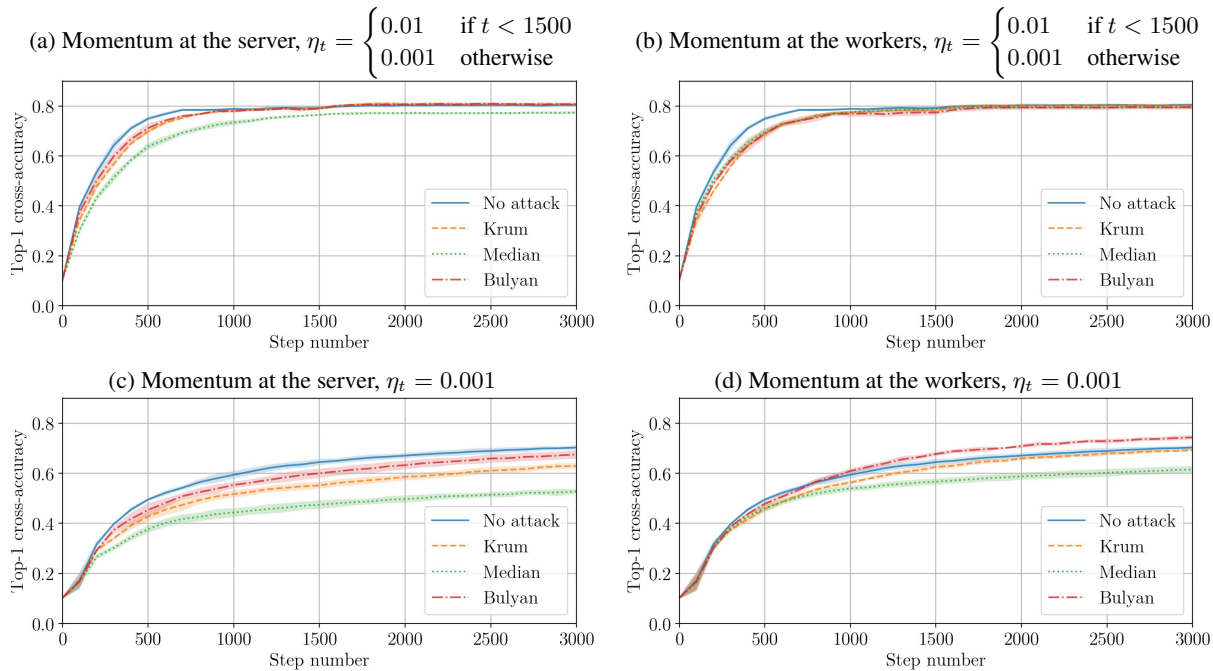


Figure 13. CIFAR-10 using  $n = 25$  workers, including  $f = 5$  Byzantine workers implementing (Xie et al., 2019). This is the maximum number of Byzantine workers *Bulyan* can support. With a reduced fraction of Byzantine workers to a quarter compared to Figure 12, the effect of the attack is almost void. Notably in this setting, *Bulyan* improves the cross-accuracy gain per step over *Krum* and *Median*.

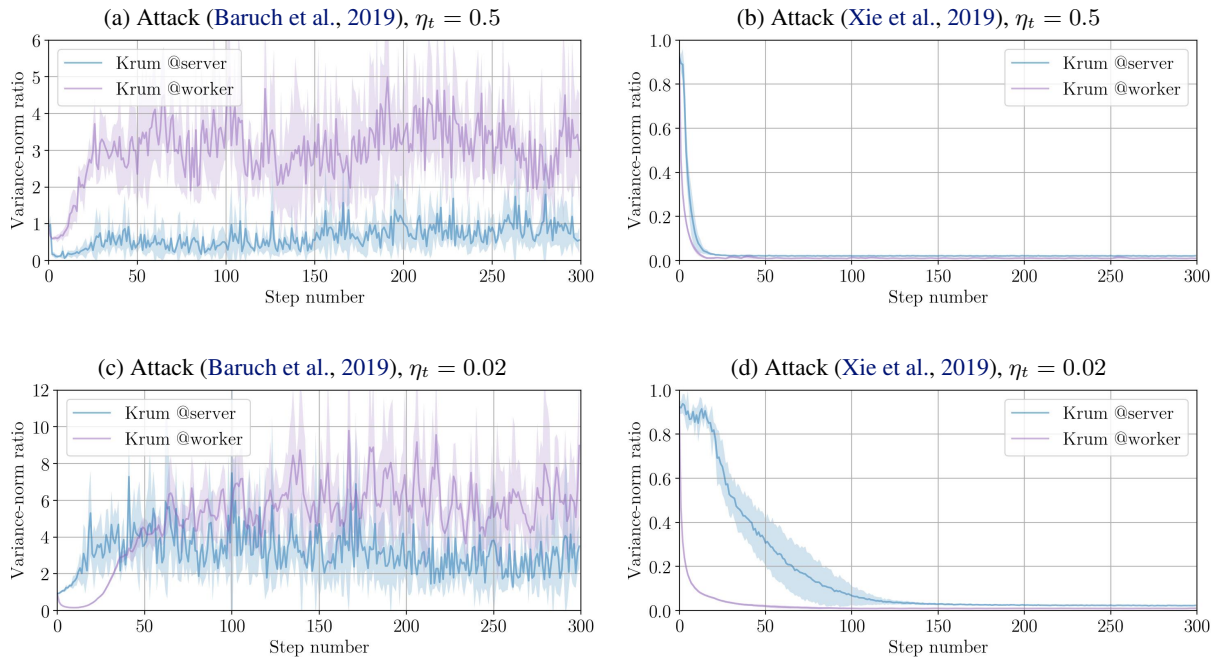


Figure 14. MNIST using  $n = 51$  workers, including  $f = 24$  Byzantine workers defended against by *Krum*. This setting contains the full range of behaviors one can observe in the subsequent figures. One first, notable behavior was predicted by the theory: until convergence is reached, reducing the learning rate decreases the variance-norm ratio. The second, recurrent behavior is that, when the model is driven useless (Figure 8), the ratio reaches low values. Such decreasing curves (14b and 14d) are empirical, distinctive signal of a very successful attack. Indeed for a successful *defense*, one should expect the ratio to grow to infinity, as the norm of the honest gradient goes toward 0.

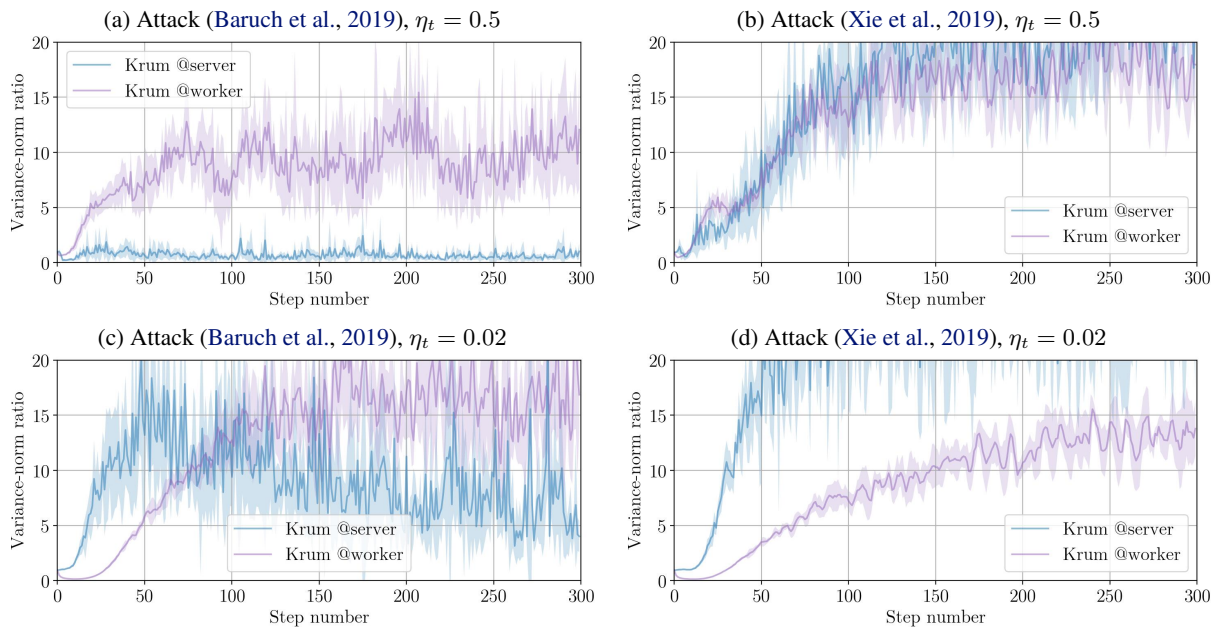


Figure 15. MNIST using  $n = 51$  workers, including  $f = 12$  Byzantine workers defended against by *Krum*. See Figure 14.



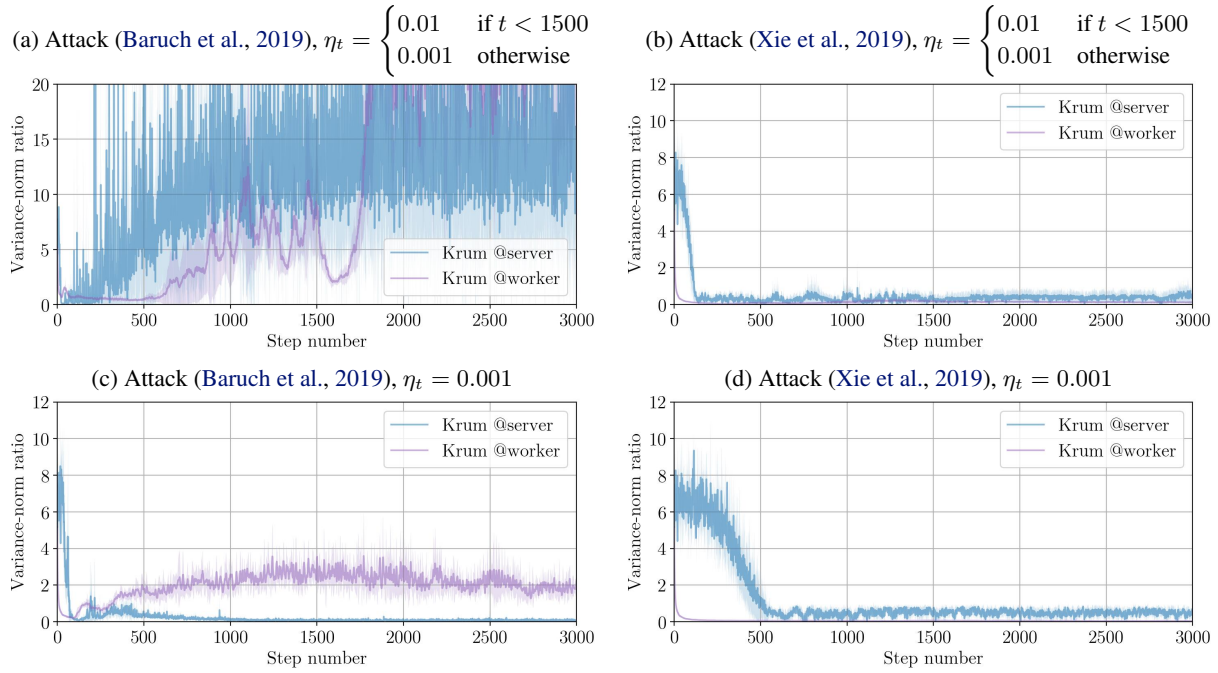


Figure 16. CIFAR-10 using  $n = 25$  workers, including  $f = 11$  Byzantine workers defended against by *Krum*. See Figure 14.

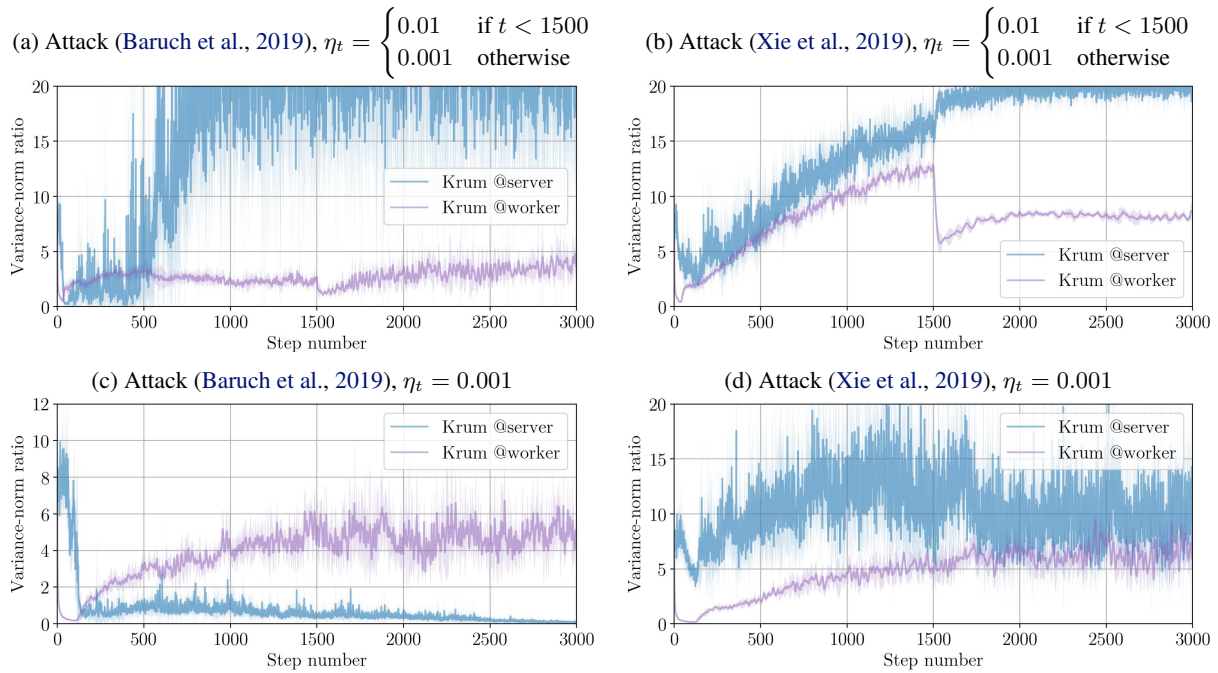


Figure 17. CIFAR-10 using  $n = 25$  workers, including  $f = 5$  Byzantine workers defended against by *Krum*. The “fracture” is due to the fact that the learning rate is decreased at step 1500. See Figure 14.

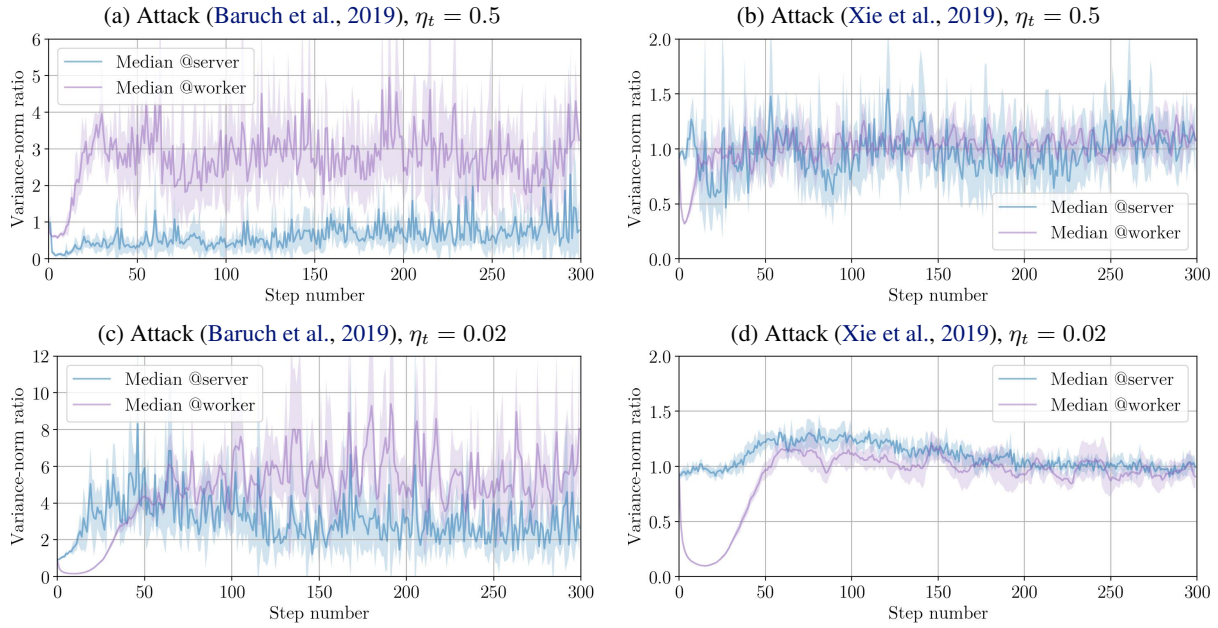


Figure 18. MNIST using  $n = 51$  workers, including  $f = 24$  Byzantine workers defended against by *Median*. See Figure 14.

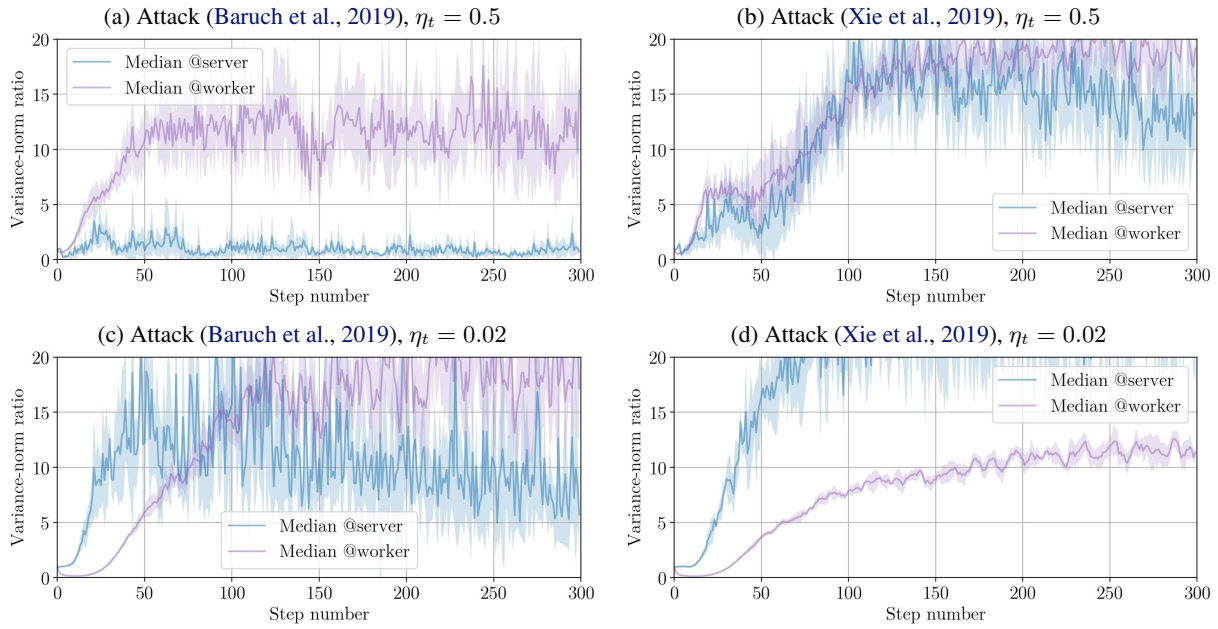


Figure 19. MNIST using  $n = 51$  workers, including  $f = 12$  Byzantine workers defended against by *Median*. See Figure 14.

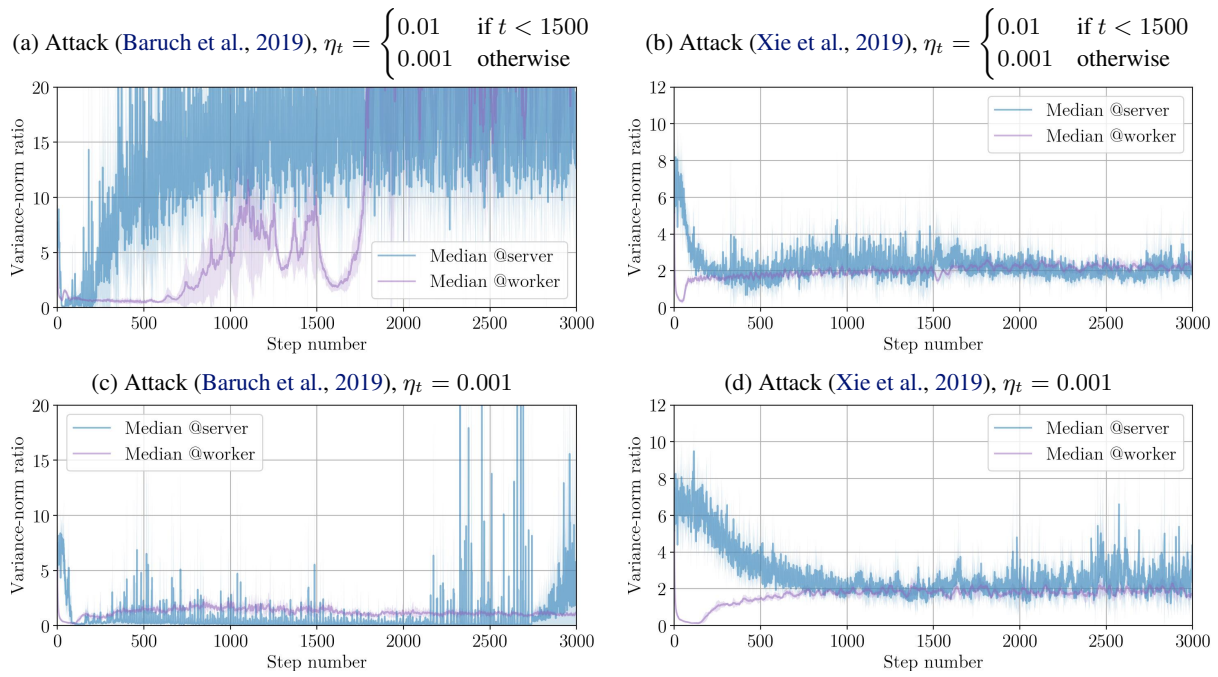


Figure 20. CIFAR-10 using  $n = 25$  workers, including  $f = 11$  Byzantine workers defended against by *Median*. See Figure 14.

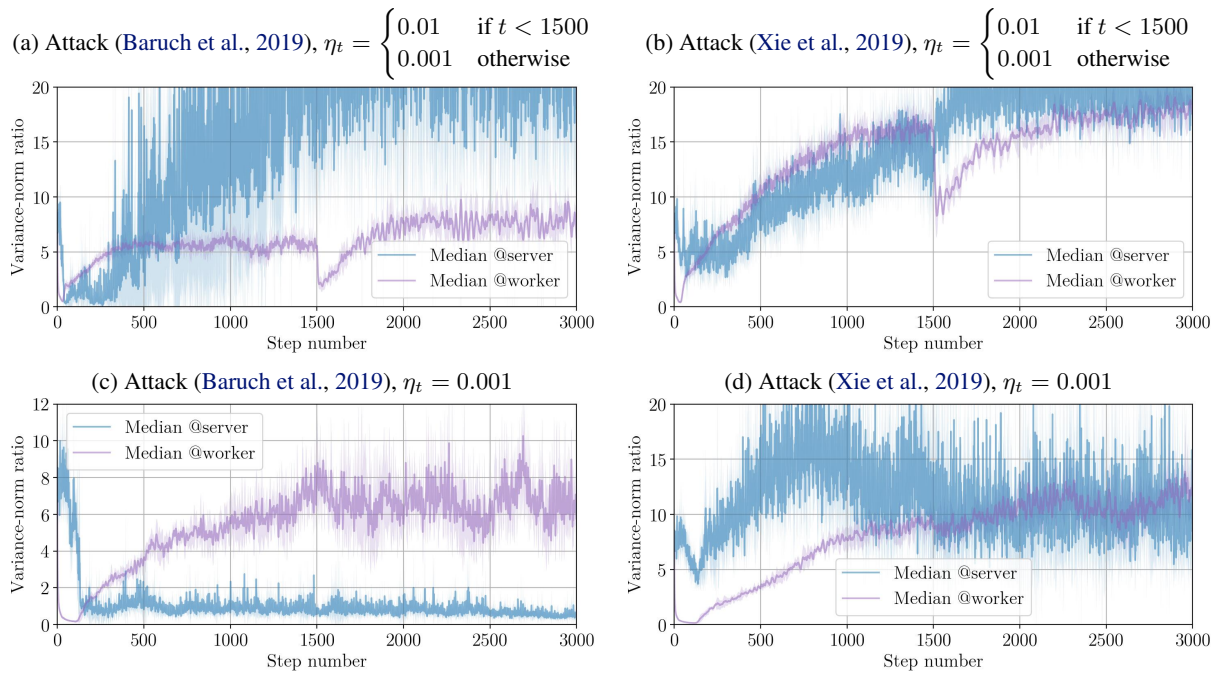


Figure 21. CIFAR-10 using  $n = 25$  workers, including  $f = 5$  Byzantine workers defended against by *Median*. See Figure 14.

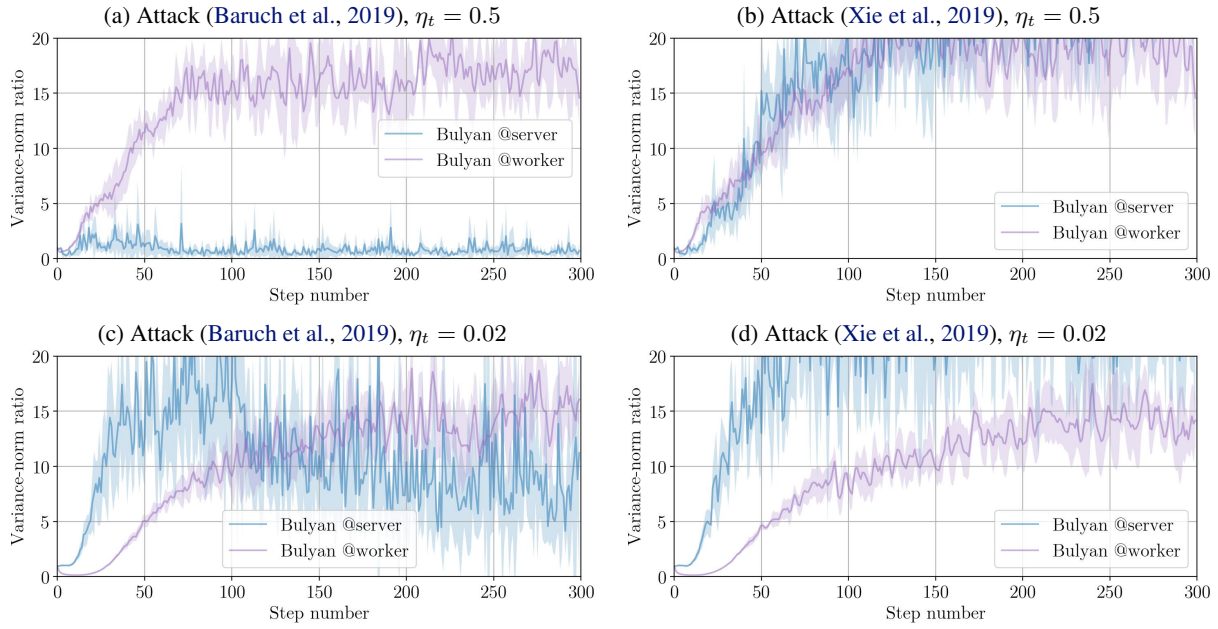


Figure 22. MNIST using  $n = 51$  workers, including  $f = 12$  Byzantine workers defended against by *Bulyan*. See Figure 14.

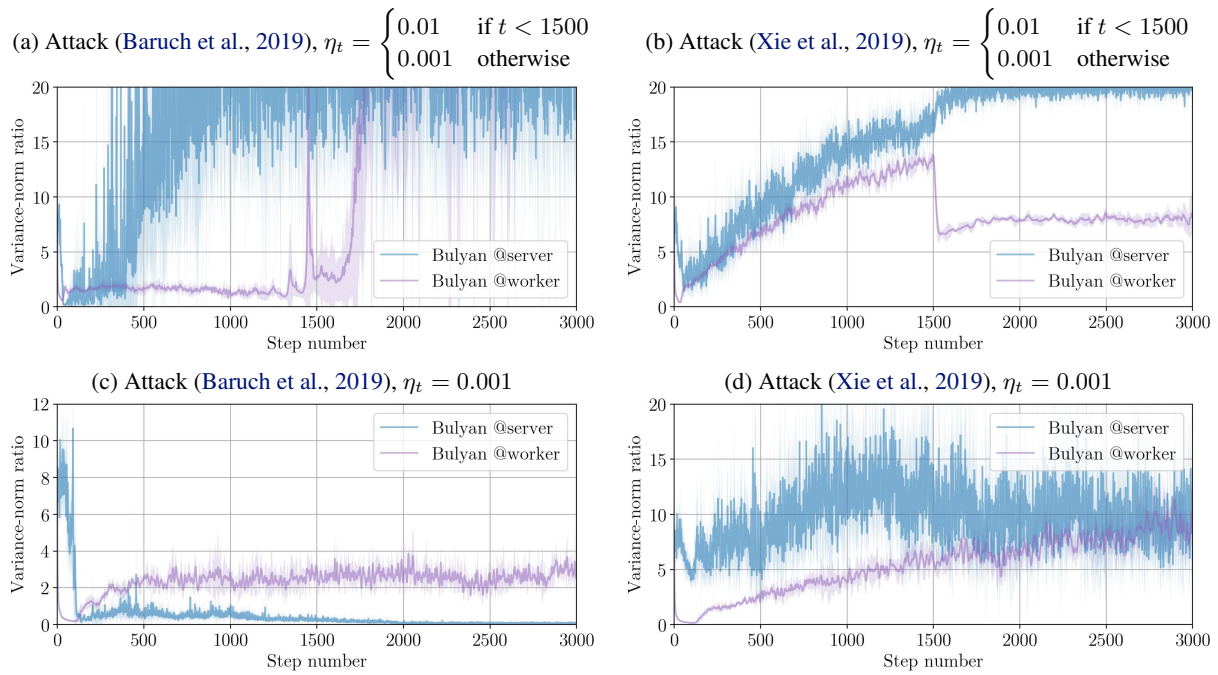


Figure 23. CIFAR-10 using  $n = 25$  workers, including  $f = 5$  Byzantine workers defended against by *Bulyan*. See Figure 14.