Distributed Computing (2022) 35:305–331
https://doi.org/10.1007/s00446-022-00427-9

Genuinely distributed Byzantine machine learning

El-Mahdi El-Mhamdi[1] · Rachid Guerraoui[1] · Arsany Guirguis[1] · Lê-Nguyên Hoang[1] · Sébastien Rouault[1]

Received: 1 December 2020 / Accepted: 11 April 2022 / Published online: 26 May 2022

Abstract
Machine learning (ML) solutions are nowadays distributed, according to the so-called *server/worker* architecture. One *server* holds the model parameters while several *workers* train the model. Clearly, such architecture is prone to various types of component failures, which can be all encompassed within the spectrum of a Byzantine behavior. Several approaches have been proposed recently to tolerate Byzantine workers. Yet all require trusting a central parameter server. We initiate in this paper the study of the *"general" Byzantine-resilient* distributed machine learning problem where no individual component is trusted. In particular, we distribute the parameter server computation on several nodes. We show that this problem can be solved in an asynchronous system, despite the presence of $\frac{1}{3}$ Byzantine parameter servers (i.e., $n_{ps} > 3f_{ps} + 1$) and $\frac{1}{3}$ Byzantine workers (i.e., $n_w > 3f_w$), which is asymptotically optimal. We present a new algorithm, *ByzSGD*, which solves the general Byzantine-resilient distributed machine learning problem by relying on three major schemes. The first, *scatter/gather*, is a communication scheme whose goal is to bound the maximum drift among models on correct servers. The second, *distributed median contraction* (DMC), leverages the geometric properties of the median in high dimensional spaces to bring parameters within the correct servers back close to each other, ensuring safe and lively learning. The third, *Minimum-diameter averaging* (*MDA*), is a statistically-robust gradient aggregation rule whose goal is to tolerate Byzantine workers. *MDA* requires a loose bound on the variance of non-Byzantine gradient estimates, compared to existing alternatives [e.g., Krum (Blanchard et al., in: Neural information processing systems, pp 118-128, 2017)]. Interestingly, *ByzSGD* ensures Byzantine resilience without adding communication rounds (on a normal path), compared to vanilla non-Byzantine alternatives. *ByzSGD* requires, however, a larger number of messages which, we show, can be reduced if we assume synchrony. We implemented *ByzSGD* on top of both TensorFlow and PyTorch, and we report on our evaluation results. In particular, we show that *ByzSGD* guarantees convergence with around 32% overhead compared to vanilla SGD. Furthermore, we show that *ByzSGD*'s throughput overhead is 24–176% in the synchronous case and 28–220% in the asynchronous case.

Keywords Distributed machine learning · Robust machine learning · Byzantine fault tolerance · Byzantine parameter servers

1 Introduction

1.1 The fragility of distributed ML

Distributing machine learning (ML) tasks seems to be the only way to cope with ever-growing datasets [16,39,49]. A common way to distribute the learning task is through the now classical *parameter server* architecture [45,46]. In short, a central *server* holds the model parameters (e.g., weights of a neural network) whereas a set of *workers* perform gradient computation (e.g., using backpropagation [35]), typically following the standard optimization algorithm: *stochastic gradient descent* (SGD) [55], on their local data, using the latest model they pull from the server. This server in turn gathers the updates from the workers, in the form of *gradients*, and aggregates them, usually through averaging [40]. This scheme is, however, very fragile because averaging does not tolerate a single corrupted input [11], whilst the multiplicity of machines increases the probability of a misbehavior somewhere in the network.

This fragility is problematic because ML is expected to play a central role in safety-critical tasks such as driving and

An early version of this paper [23] appeared in the 39th Symposium on Principles of Distributed Computing (PODC), 2020.

✉ Arsany Guirguis
arsany.guirguis@epfl.ch

1 School of Computer and Communication Sciences (IC), Ecole Polytechnique Fédérale de Lausanne (EPFL), Route Cantonale, 1015 Lausanne, VD, Switzerland

flying people, diagnosing their diseases, and recommending treatments to their doctors [25,43]. Little room should be left for the routinely reported [10,29,50] vulnerabilities of today's ML solutions.

1.2 Byzantine-resilient ML

Over the past few years, a growing body of work, e.g., [4,8,11, 15,20,60,63,67], took up the challenge of *Byzantine-resilient ML*. The Byzantine failure model, as originally introduced in distributed computing [42], encompasses crashes, software bugs, hardware defects, message omissions, corrupted data, and even worse, hacked machines [9,64]. So far, all the work on Byzantine-resilient ML assumed that a fraction of workers could be Byzantine. But all assumed the central parameter server to be always *honest* and *failure-free*.[1] In other words, none of the previous approaches considered a genuinely Byzantine-resilient distributed ML system, in the distributed computing sense, where no component is trusted. Consider a multi-branch organization with sensitive data, e.g., a hospital or a bank, that would like to train an ML model among its branches. In such a situation, the worker machines, as well as the central server, should be robust to the worst: the adversarial attacks.

A natural way to prevent the parameter server from being a *single point-of-failure* is to *replicate* it. But this poses the problem of how to synchronize the replicas. The classical synchronization technique, *state machine replication* [13,56] (SMR), enforces a *total order for updates* on all replicas through *consensus*, providing the abstraction of a single parameter server, while benefiting from the resilience of the multiplicity of underlying replicas. Applying SMR to distributed SGD would however lead to a potentially huge *overhead*. In order to maintain the same state, replicas would need to agree on a total order of the model updates, inducing frequent exchanges (and retransmissions) of gradients and parameter vectors, that can be several hundreds of MB large [39]. Given that a distributed ML setup is network bound [36,70], SMR is impractical in this context.

The key insight underlying our paper is that the *general Byzantine SGD problem*, even when neither the workers nor the servers are trusted, is easier than consensus (which is based on SMR in the distributed computing sense), i.e., total ordering of updates is not required in the context of ML applications; only convergence to a good final accuracy is needed. We thus follow a different route where we do not require all the servers to maintain the same state . Instead, we allow mildly diverging parameters (which have proven beneficial in other contexts [5,71]) and present new ways to *contract* them in a *distributed* manner.

1.3 Contributions

In short, we consider, for the first time in the context of distributed SGD, the general Byzantine ML problem, where no individual component is trusted, in the context of i.i.d. local data distributions. First, we show that this problem can be solved in an asynchronous setting. We then show how we can utilize synchrony to boost the performance of our asynchronous solution.

Our main algorithm, *ByzSGD*, achieves *general* resilience without assuming bounds on communication delays, nor inducing additional communication rounds (on a normal path), when compared to a non-Byzantine resilient scheme. We prove that *ByzSGD* tolerates $\frac{1}{3}$ Byzantine servers (i.e., $n_{ps} > 3f_{ps} + 1$) and $\frac{1}{3}$ Byzantine workers (i.e., $n_w > 3f_w$), which is asymptotically optimal in an asynchronous setting. *ByzSGD* employs a novel communication scheme, *Scatter/Gather*, that bounds the maximum drift between models on correct servers. In the *scatter* phase, servers work independently (they do not communicate among themselves) and hence, their views of the model could drift away from each other. In the *gather* phase, correct servers communicate and apply collectively a *Distributed Median-based Contraction* (DMC) module. This module is crucial for it brings the diverging parameter vectors back closer to each other, despite each parameter server being only able to gather a fraction of the parameter vectors. Interestingly, *ByzSGD* ensures Byzantine resilience without adding communication rounds (on a normal path), compared to non-Byzantine alternatives. *ByzSGD* requires, however, a larger number of messages which we show we can reduce if we assume synchrony. Essentially, in a synchronous setting, workers can use a novel filtering mechanism we introduce to eliminate replies from Byzantine servers without requiring to communicate with all servers.

ByzSGD[2] uses a *statistically-robust gradient aggregation rule* (GAR), which we call *Minimum-diameter averaging* (*MDA*) to tolerate Byzantine workers. Such a choice of a GAR has two advantages compared to previously-used GARs in the literature to tolerate Byzantine workers, e.g., [11,65]. First, *MDA* requires a loose bound on the variance of the non-Byzantine gradient estimates, which makes it practical.[3] Second, *MDA* makes the best use of the variance

[1] Some works, e.g., [34], used multiple servers to do multi-party computation (MPC) to solve the problem of untrusted servers. Although MPC achieves privacy, it is not Byzantine-resilient: it tolerates honest-but-curious servers but not Byzantine (e.g., malicious) servers.

[2] We use the same name for our asynchronous and synchronous variants of the algorithm when there is no ambiguity.

[3] In Sect. 7, we report on our experimental validation of such a requirement. We compare the bound on the variance require by *MDA* to the one required by *Multi-Krum* [11], a state-of-the-art GAR. For instance,

reduction resulting from the gradient estimates on multiple workers, unlike Krum [11] and Median [65].

We prove that *ByzSGD* guarantees learning liveness and safety despite the presence of Byzantine machines, be they workers or servers. We implemented *ByzSGD* on top of TensorFlow [1] and PyTorch [51], while achieving transparency: applications implemented with either framework need not to change their interfaces to be made Byzantine-resilient. We also report on our evaluation of *ByzSGD*. We show that *ByzSGD* tolerates Byzantine failures with a reasonable convergence overhead ($\sim$ 32%) compared to the non Byzantine-resilient vanilla SGD deployment. Moreover, we show that the throughput overhead of *ByzSGD* ranges from 24 to 220% compared to vanilla SGD. The code used in our experiments is based on *Garfield* [31], which is open-sourced and accessible from https://github.com/LPD-EPFL/Garfield.

The paper is organized as follows. Section 2 provides some background on SGD and Byzantine resilience. Section 3 describes the problem settings and the threat model. Section 4 describes our *ByzSGD* algorithm. Section 5 depicts its correctness proof. Section 6 discusses how *ByzSGD* leverages synchrony to boost performance. Section 7 reports on our empirical evaluation of *ByzSGD*. Section 8 concludes the paper by discussing related work and highlighting open questions.

2 Background

2.1 Stochastic gradient descent

Stochastic gradient descent (SGD) [55] is a widely-used *optimization* algorithm in ML applications [1,16,45]. Typically, SGD is used to minimize a *loss function* $L(\theta) \in \mathbb{R}$, which measures how accurate the model θ is when classifying an input. Formally, SGD addresses the following optimization problem:

$$\underset{\theta \in \mathbb{R}^d}{\arg\min}\, L(\theta) \tag{1}$$

SGD works iteratively and consists, in each step t, of:

1. Estimating the gradient $\nabla L\left(\theta^{(t)}, \xi\right)$, with a subset ξ of size b of the training set, called *mini-batch*. Such a gradient is a *stochastic* estimation of the real, uncomputable one $\nabla L\left(\theta^{(t)}\right)$.

(Footnote 3 continued) we show that with a batch-size of 100 and assuming 1 Byzantine failure, the requirement of *MDA* is satisfied in our experiments, while that of *Multi-Krum* is not.

2. Updating the parameters following the estimated gradient:

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \cdot \nabla L\left(\theta^{(t)}, \xi\right) \tag{2}$$

The sequence $\{\eta_t \in \mathbb{R}_{>0}\}$ is called the *learning rate*.

2.2 The parameter server architecture

Estimating one gradient is computationally expensive, as it consists in computing b estimates of $\nabla L(\theta, \xi_i)$, where ξ_i is the ith pair (input, label) from the mini-batch, and where each $\nabla L(\theta, \xi_i)$ might involve one *backpropagation* computation [35] (in the case of training a neural network). Hence, the amount of arithmetic operations to carry out to estimate $\nabla L(\theta, \xi) \approx \nabla L\left(\theta^{(t)}\right)$ is $\mathcal{O}(b \cdot d)$.

However, this gradient estimation can be easily distributed: the b computations of $\nabla L\left(\theta^{(t)}, \xi_i\right)$ can be executed in parallel on n machines, where the aggregate of such computations gives $\nabla L\left(\theta^{(t)}, \xi\right)$. This corresponds to the now standard *parameter server* architecture [46], where a central server holds the parameters θ.

Each training step includes 2 communication rounds: the server first broadcasts the parameters to workers, which then estimate the gradient $\nabla L\left(\theta^{(t)}, \xi_i\right)$ (i.e., each with a mini-batch of $\frac{b}{n}$). When a worker completes its estimation, it sends it back to the parameter server, which in turn *averages* these estimations and updates the parameters θ, as in Eq. 2.

2.3 Byzantine machine learning

The Byzantine failure abstraction [42] models any arbitrary behavior and encompasses software bugs, hardware defects, message omissions, or even hacked machines. We then typically assume that a subset of the machines can be Byzantine and controlled by an adversary whose sole purpose is to defeat the computation.

A Byzantine machine can, for instance, send a biased estimate of a gradient to another machine, which leads to a corrupted learning model accordingly or even to learning divergence [7]. Byzantine failures also abstract the *data poisoning* problem [9], which happens when a machine owns maliciously-labeled data. This may result in learning a corrupted model, especially-crafted by the adversary. Clearly, assuming a central machine controlling the learning process (as with the standard parameter server architecture [46]) is problematic if such a machine is controlled by an adversary, for this machine can write whatever it wants to the final model, jeopardizing the learning process.

2.3.1 Byzantine-resilient aggregation

Robust aggregation of gradients is the key for Byzantine workers' resilience. To this end, gradients are processed by a gradient aggregation rule (GAR), which purpose is to ensure that output of aggregation is as close as possible to the real gradient of the loss function.

In the general theory of stochastic gradient descent (SGD) convergence, a typical validity assumption is that the gradient estimator is unbiased [12]. The role of a GAR is to ensure a relaxed version of this assumption in order to accommodate for the presence of malicious workers (whose gradients are potentially biased).

Definition 1 gives such a relaxation, which we adapt from [11,24] and which was used as a standard for Byzantine resilience, either explicitly, e.g. in [33,34,37,52,57,65–67,69] or implicitly, by replacing the angle criterion by a distance criterion, e.g. in [47].

Definition 1 Let $0 \leq \alpha < \frac{\pi}{2}$ be any angular value and $0 \leq f \leq q$ with q the total number of input vectors to the GAR and f the maximum number of Byzantine vectors. Let g be an unbiased estimate of the true gradient ∇L (computed by at least $q - f$ honest workers), i.e., $\mathbb{E}g = \nabla L$.

A GAR (whose output noted as $\mathcal{F}$) is robust (said to be (α, f)-Byzanitne resilient) if
$\langle \mathbb{E}\mathcal{F}, \nabla L \rangle \geq (1 - \sin\alpha) \cdot \|\nabla L\|_2^2 > 0$.

Briefly, Definition 1 ensures that the aggregated gradient (using $\mathcal{F}$) is not very far from the true gradient ∇L, i.e., the angle between both is at most α. We follow the same notion of (α, f)-Byzantine resilience in this paper, i.e., we show that *ByzSGD* is (α, f)-Byzantine resilient.

2.4 *ByzSGD*'s GARs

ByzSGD uses two GARs: *Minimum-diameter averaging*[4] and coordinate-wise *Median* [65]. Note that although GARs are typically used to aggregate gradients (as suggested by the name), we use the coordinate-wise *Median* to aggregate models. Simply, in this case, the input is a set of models rather than a set of gradients. In this section, we present these two GARs in detail.

2.4.1 Minimum diameter averaging (MDA)

MDA is a GAR that ensures resilience against a minority of Byzantine input gradients. Mathematically, this function was introduced in [54] and its Byzantine resilience proof was given in [24]. *MDA* satisfies the (α, f) Byzantine resilience guarantees introduced in [11]. Formally, let $\mathcal{X}$ be the set of input gradients (all follow the same distribution), out of them f are Byzantine (with $|\mathcal{X}| = q$), and y be the output of the Byzantine resilient GAR. Then, the following properties hold:

1. $\mathbb{E}y$ is in the same half-space as $\mathbb{E}\mathcal{X}$, and
2. the first 4 statistical moments of y are bounded above by a linear combination of the first 4 statistical moments of $x \sim \mathcal{X}$.

Such conditions are sufficient to show that the output of this GAR guarantees convergence of the learning procedure. More formally, these conditions enable the GAR to have a proof that follows from the *global confinement* proof of SGD [12].

In order to work, *MDA* assumes the following:

$$\exists \kappa \in]1, +\infty[, \forall \left(i, t, \theta\right) \in [1 \ldots q - f] \times \mathbb{N} \times \mathbb{R}^d,$$
$$\kappa \frac{\sqrt{8}f}{q - f} \sqrt{\mathbb{E}\left(\left\|g_t^{(i)} - \mathbb{E}g_t^{(i)}\right\|_2^2\right)} \leq \left\|\nabla L\left(\theta\right)\right\|_2, \tag{3}$$

where θ is the model state at the training step t, q is the total number of input gradients, f is the maximum number of Byzantine gradients, g_t is an unbiased estimate of the gradient at step t, and L is the loss function. Note that towards convergence to a local minimum, such a condition is not guaranteed to hold. Yet, this is an evident limitation for all similar GARs that are used in the ML context. For example, Krum [11] and Multi-Krum [17] guarantee convergence only to a ball around the local minimum. Moreover, El-Mhamdi et al. have shown in [22] that this condition is necessary for Byzantine resilience, i.e., it is impossible to guarantee convergence if this condition is not satisfied.

The *MDA* function works as follows. Consider that the total number of gradients is q and the maximum number of Byzantine gradients is f with $q \geq 2f + 1$. *MDA* enumerates all subsets of size $q - f$ from the input gradients and finds the subset with the *minimum diameter* among all subsets of this size, i.e., $q - f$. The *diameter* of a subset is defined as the maximum distance[5] between any two elements of this subset. The output of the *MDA* function is the average of gradients in such a subset. More formally, the *MDA* function is defined as follows:

Let $\left(g_1 \ldots g_q\right) \in \left(\mathbb{R}^d\right)^q$, and $\mathcal{X} \triangleq \left\{g_1 \ldots g_q\right\}$ the set containing all the input gradients.

Let $\mathcal{R} \triangleq \{\mathcal{Q} \mid \mathcal{Q} \subset \mathcal{X}, |\mathcal{Q}| = q - f\}$ the set of all the subsets of $\mathcal{X}$ with a cardinality of $q - f$, and let:

$$\mathcal{S} \triangleq \underset{\mathcal{Q} \in \mathcal{R}}{\arg\min} \left(\max_{(g_i, g_j) \in \mathcal{Q}^2} \left(\left\|g_i - g_j\right\|_2\right) \right).$$

[4] Basically, any GAR that satisfies such a form of resilience, e.g. [11, 24,65], can be used with *ByzSGD*; *MDA* is just an instance.

[5] In this paper, we always consider the ℓ_2 distances.

Then, the aggregated gradient is given by:

$$MDA\left(g_1 \ldots g_q\right) \triangleq \frac{1}{q-f} \sum_{g \in \mathcal{S}} g.$$

Notably, *MDA* carries an exponential asymptotic complexity of $\mathcal{O}\left(\binom{q}{f} + q^2 d\right)$. Yet, its assumptions about variance (Eq. (3)) are weaker than other GARs including *Median* and Multi-Krum.

2.4.2 Coordinate-wise *Median*

We formally define the *median* as follows:

$$\forall q \in \mathbb{N} - \{0\}, \forall \left(x_1 \ldots x_q\right) \in \mathbb{R}^q, \quad median\left(x_1 \ldots x_q\right) \triangleq x_s \in \mathbb{R}$$

such that with *sorted* $\left(x_1 \ldots x_q\right)$,

$$\begin{cases} x_s = x_{\lceil \frac{q}{2} \rceil} & \text{if } q \text{ is odd} \\ x_s = \frac{x_{\frac{q}{2}} + x_{\frac{q}{2}+1}}{2} & \text{if } q \text{ is even,} \end{cases}$$

where q is the total number of input gradients.

We formally define the coordinate-wise median as follows:

$$\forall (d, q) \in (\mathbb{N} - \{0\})^2, \forall \left(x_1 \ldots x_q\right) \in \left(\mathbb{R}^d\right)^q, \quad median\left(x_1 \ldots x_q\right) \triangleq x_s \in \mathbb{R}^d$$

such that:

$$\forall i \in [1 \ldots d], x_s[i] = median\left(x_1[i] \ldots x_q[i]\right)$$

Coordinate-wise median, hereafter simply: *Median*, requires $q > 2f + 1$ and its Byzantine resilience was proved in [65].

3 System model

3.1 Overview

We build on the standard parameter server model, with two main variations (Fig. 1).

1. We assume a subset of the nodes (i.e., machines) involved in the distributed SGD process to be *adversarial*, or *Byzantine*, in the parlance of [11,15,24]. The other nodes are said to be *correct*. We denote such nodes by h, i.e., $h_w = n_w - f_w$ and $h_{ps} = n_{ps} - f_{ps}$ (see Table 1 for all the notations). Clearly, which node is correct and which is Byzantine is not known ahead of time.
2. We consider several *replicas* of the parameter server (we call them *servers*), instead of a single one, preventing it from being a single *point-of-failure*, unlike in classical ML approaches.

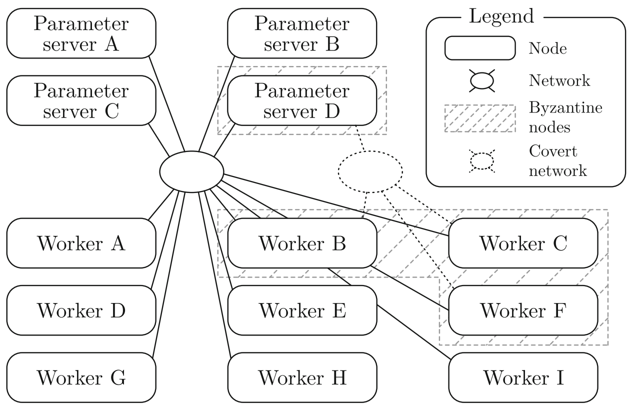


Fig. 1 A distributed ML setup with 4 parameter servers and 9 workers, including respectively 1 and 3 Byzantine nodes, which all can be viewed as a single adversary

In our new context, workers send their gradients to all servers (instead of one in the standard case), which in turn send their parameters to all workers. Periodically, servers communicate with each other, as we describe later in Sect. 4. We consider *bulk-synchronous training*: only the gradients computed at a learning step t are used for the parameters update at the same step t. This upper-bounds the space complexity at any node by $\mathcal{O}(n_w d)$ (assuming $n_w \geq n_{ps}$). Put differently: if a node is executing learning at step t, it discards any result obtained (by itself or by any other node) at step $< t$. This is safe for both workers and parameter servers. For example, if an honest worker w_1 is stuck at iteration t_0, due to its slowness, while the majority of honest workers managed to execute iterations up to t_1 (with $t_1 >> t_0$), w_1 does not need to store intermediate models and/or gradients of all steps $\in [t_0, t_1]$. Instead, w_1 should only fetch the latest model (of t_1) from a majority of honest parameter servers. The same goes for a parameter server ps_1, who may be at iteration t_0, and receive gradients of iterations $t_1 > t_0$. In this case, ps_1 may then safely skip its current iteration and move to iteration t_1. We consider an asynchronous network: there is no upper bound on the time it takes to receive a message from other nodes in the system.

3.1.1 Adversary capabilities

The adversary is an entity that controls all Byzantine nodes (Fig. 1), and which goal is to prevent the SGD process from converging to a state that could have been achieved if there

Table 1 The notations used throughout this paper

n_{ps}	Total number of parameter servers
f_{ps}	Declared, maximal number of Byzantine parameter servers
q_{ps}	Number of parameter vectors a node waits for from servers, $2f_{ps} + 2 \leq q_{ps} \leq n_{ps} - f_{ps}$
n_w	Total number of workers
f_w	Declared, maximal number of Byzantine workers
q_w	Number of gradients a node waits for from workers, $2f_w + 1 \leq q_w \leq n_w - f_w$
d	Dimension of the parameter space $\mathbb{R}^d$
L	Loss function we aim to minimize
l	Lipschitz constant of the loss function
$\theta_t^{(i)}$	Parameter vector (i.e. model) at the parameter server i at step t
$\mathcal{G}_t^{(i)}$	Gradient distribution at the worker i at step t
$g_t^{(i)}$	Stochastic gradient estimation of worker i at step t
$\nabla L\left(\theta\right)$	Real gradient of the loss function L at θ
$G_t^{(i)}$	Aggregated gradient of worker i at step t
$\theta_{t+1/2}^{(i)}$	Parameter vector of server i, right after update using $G_t^{(i)}$
$\hat{G}_t^{(i)}$	Effective gradient, combining scatter and gather
η_t	Learning rate at step t
$\xi_t^{(i)}$	Gradient estimation error, i.e., $\xi_t^{(i)} = g_t^{(i)} - \nabla L\left(\theta\right)$

Without loss of generality in the analysis, we note:
$\left[1 \ldots n_{ps} - f_{ps}\right]$ the indexes of the correct servers
$[1 \ldots n_w - f_w]$ the indexes of the correct workers

was no adversary. As in [11,15,24], we assume an omniscient adversary that can see the full training datasets as well as the packets being transferred over the network (i.e., gradients and models). However, the adversary is not omnipotent: it can only send arbitrary messages from the nodes it controls, or force them to remain silent. We assume nodes can authenticate the source of a message, so no Byzantine node can forge its identity or create multiple fake ones [14]. Note that there is a tradeoff between the adversary power and the resilience cost. In this paper, we assume that the strongest adversary can only hack a minority of the machines and cannot control the network. We believe this is reasonable in some practical scenarios. For example, assume a multi-branch bank which train a model to predict future investments. Each branch is having its own independent security system, where the network communication is provided by a trusted third party. In this case, an adversary can hack some of these security systems of the independent branches (hopefully only a minority) without being able to hack the network provider. The same argument can be made for multiple organizations which decide to solve a similar problem together via training some ML model on their local data.

Let $\zeta_t \times (k)^{(a)}$ denote the subset of size k of some set $\left\{\zeta_t^{(1)} \ldots \zeta_t^{(q)}\right\}$ delivered by node a at step t. To highlight the fact that such a subset can contain up to f arbitrary (Byzantine) vectors, we will also denote it by $\left(\zeta_t \times (k-f)^{(a)}, \zeta_t \times (f)^{(a)}\right)$. The exact value of q depends on the context: in the proof, it will always be q_w when $\zeta_t^{(i)}$ denotes a gradient, and q_{ps} otherwise.

3.2 Byzantine resilience conditions

We follow the classical conditions for convergence in non-convex optimization [12]. For instance, we assume that the training data is identically and independently distributed ($i.i.d$) over the workers, and gradient estimations at honest workers, i.e., $\forall t \in \mathbb{N}$, $g_t^{(1)} \ldots g_t^{(n_w - f_w)}$ are mutually independent. Moreover, we assume that such estimations have a bounded standard deviation:

$$\exists \sigma' \in \mathbb{R}_+, \forall (i, t) \in [1 \ldots n_w - f_w] \times \mathbb{N},$$
$$\mathbb{E}\left\|g_t^{(i)} - \mathbb{E}g_t^{(i)}\right\|_2 \leq \sigma'. \tag{4}$$

Note that to realize this assumption, the mini-batch size should be lower-bounded, i.e., $\sim \Omega\left(\sqrt{d}\right)$. Previous approaches, e.g., [38], theoretically proved that leveraging momentum reduces variance and prevents Byzantine attacks without increasing the batch size Furthermore, typical variance reduction techniques, e.g., [3], can be employed to guarantee Eq. 4. The upper-bound on the estimate of the gradient is constant, i.e., we do not assume vanishing gradient error. These estimates, of the true gradient, are unbiased with sufficiently low variance (to satisfy *MDA*'s condition, namely Eq. 3).

Such an equation bounds the ratio of the standard deviation of the stochastic gradient estimations to the norm of the real gradient with $\frac{h_w}{\sqrt{8 f_w}}$. This assumption is now classical in the Byzantine ML literature [11,34,52,65]. We also empirically verify this assumption in our experimental setup (see Sect. 7).

Let η_t be the learning rate at the learning step $t \in \mathbb{N}$ with the following specifications:

1. The sequence of learning rates η_t is decreasing[6] with t, i.e., if $t_a > t_b$ then, $\eta_{t_a} < \eta_{t_b}$. Thus, the initial learning rate η_1 is the largest value among learning rates used throughout the learning phase. In fact, to prove Byzantine resilience in Sect. 5, we require $\eta_t = 1/t^\alpha$, with $\alpha \in]\frac{1}{2}, 1]$.
2. The sequence of learning rates η_t satisfies $\sum_t \eta_t = \infty$ and $\sum_t {\eta_t}^2 < \infty$.

We assume the loss function L to be positive, 3-times differentiable with continuous derivatives, and l-Lipschitz continuous, i.e.,

$$\exists l > 0, \quad \forall (x, y) \in \left(\mathbb{R}^d\right)^2, \\ \left\| \nabla L(x) - \nabla L(y) \right\|_2 \leq l \left\| x - y \right\|_2 . \tag{5}$$

Such standard l-Lipschitz continuity assumption [6,53,62] acts as the only bridge between the parameter vectors $\theta_t^{(i)}$ and the stochastic gradients $g_t^{(i)}$ at a given step t. This is a *liveness* assumption: the value of l can be arbitrarily high and is only used to bound the expected maximal distance between any two correct parameter vectors.

We also follow the standard assumptions of [11,12] as follows. We assume (1) bounded statistical moments $\forall r \in [2 \ldots 4], \exists (A_r, B_r) \in \mathbb{R}^2, \forall \left(i, t, \theta\right) \in [1 \ldots n_w - f_w] \times \mathbb{N} \times \mathbb{R}^d, \mathbb{E} \left\| g_t^{(i)} \right\|_2^r \leq A_r + B_r \left\| \theta \right\|_2^r$, and (2) convexity of loss beyond a certain horizon:

$$\exists D \in \mathbb{R}, \forall \theta \in \mathbb{R}^d, \|\theta\|_2^2 \geq D, \\ \exists (\varepsilon, \beta) \in \mathbb{R}_+ \times \left[0, \frac{\pi}{2}\right[, \|\nabla L(\theta)\|_2 \geq \varepsilon, \\ \langle \theta, \nabla L(\theta) \rangle \geq \cos(\beta) \; \|\theta\|_2 \; \|\nabla L(\theta)\|_2 .$$

The latter assumption was first adapted from [12] by [11,18,24] to account for Byzantine resilience. It intuitively says that beyond a certain horizon, the loss function is "steep enough" (lower bounded gradient) and "convex enough" (lower bounded angle between the gradient and the parameter vector). The loss function does not need to be convex, but adding regularization terms such as $\|\theta\|^2$ ensures such an assumption, since close to infinity, the regularization dominates the rest of the loss function and permits the gradient $\nabla L(\theta)$ to point to the same half space as θ. The original assumption of [12] is that $\left\langle \theta, \nabla L\left(\theta\right) \right\rangle > 0$; in [11,24] it was argued that requiring this scalar product to be strictly positive is the same as requiring the angle between θ and $\nabla L\left(\theta\right)$ to be lower bounded by an acute angle ($\beta < \pi/2$).

Regarding the network, we assume no (pattern of) network partitioning among the correct servers lasts forever. This ensures that eventually the correct parameter servers can communicate with each other in order to pull their views of the model back close to each other; this is crucial to achieve Byzantine resilience. Formally, consider $\mathcal{S}$ the probability distribution over delivering configurations, and $X \sim \mathcal{S}$ a random set that follows the probability distribution $\mathcal{S}$. We assume that

$$\forall s \in S, P(X = s) > 0. \tag{6}$$

We denote $\rho = \min_{s \in S} P(X = s)$ the probability of least likely configuration. Since the set S of configurations is finite, our key assumption equivalently says that $\rho > 0$. Based on that definition, we assume for any server j, $\left\| \nabla L\left(\theta_t^{(j)}\right) \right\|_2 \geq 36 h_w \sigma' + \frac{4Cdl^2}{\rho}$, for some constant C.

Finally, *ByzSGD* requires $n_w \geq 3 f_w + 1$ and $n_{ps} \geq 3 f_{ps} + 2$. This is *asymptotically* optimal. El-Mhamdi et al. proved in [22] that Byzantine resilience is impossible with $n \leq 3f$ in asynchronous networks. The proof relies on the fact that in asynchronous networks, machines should wait for only $q = n - f$ replies and hence, we have $q \leq 2f$. With f Byzantine nodes, we can expect at most f honest replies. In this case, it is impossible to distinguish between honest and Byzantine replies and hence, Byzantine resilience is not guaranteed. Note that this result holds even with *i.i.d* data as in our case.

Some works, e.g., [32,67] in the literature claimed that Byzantine resilience can be achieved with arbitrary number of failures, i.e., with $0 \leq f < n$. The main idea is to let each worker train a local model with its local data to minimize Eq. 1, and then to report the trained model to the parameter servers, which can then apply majority voting to choose the correct model. Such idea will work only with convex optimization functions, i.e., with one unique minimizer. Since we aim for non-convex functions, local training for workers might lead to different local minima, and it will be impossible to differentiate between honest and Byzantine models, i.e., Byzantine resilience cannot be guaranteed.

4 *ByzSGD*: general Byzantine resilience

We present here *ByzSGD*, the first algorithm to tolerate Byzantine workers and servers without making any assump-

[6] In fact, it is sufficient that the sequence is decreasing only once every T steps, with $T = \frac{1}{3.l.\eta_1}$ where l is the Lipschitz constant of the loss function.

tions on node relative speeds and communication delays. *ByzSGD* does not add, on the normal path, any communication rounds compared to the standard parameter server communication model (Sect. 2.2). However, periodically, *ByzSGD* adds one communication round between servers to enforce contraction, as we show in this section.

We first describe the fundamental technique to tolerate Byzantine servers: *Distributed Median-based Contraction* (DMC). Then, we explain the overall functioning of *ByzSGD*, highlighting our novel *Scatter/Gather* communication scheme. *ByzSGD* uses *MDA* to tolerate Byzantine workers.

4.1 Distributed median-based contraction

The fundamental problem addressed here is induced by the multiplicity of servers and consists of bounding the drift among correct parameter vectors $\theta_t^{(1)} \ldots \theta_t^{(n_{ps}-f_{ps})}$, as t grows. The problem is particularly challenging because of the combination of three constraints: (a) we consider a Byzantine environment, (b) we assume an asynchronous network, and (c) we do not want to add communication rounds, compared to those done by vanilla non-Byzantine deployments, given the expensive cost of communication in distributed ML applications [36,70]. The challenging question can then be formulated as follows: given that the correct parameter servers should not expect to receive more than $n - f - 1$ messages per communication round, how to keep the correct parameters close to each other, knowing that a fraction of the received messages could be Byzantine?

Our solution to this issue is, what we call, *Distributed Median-based Contraction (DMC)*, which goal is to decrease the expected maximum distance between any two honest parameter vectors (i.e., *contract* them). DMC consists of (1) the application of *coordinate-wise* Median (which is Byzantine-resilient as soon as $q_{ps} \geq 2f_{ps}+1$) on the parameter vectors and (2) the over-provisioning of 1 more correct parameter server (i.e., $q_{ps} \geq 2f_{ps} + 2$); both constitute the root of what we call the *contraction effect*. Assuming each honest parameter server can deliver a subset of $q_{ps} - f_{ps} - 1$ *honest* parameters, the expected median of the gathered parameters is then both (1) *bounded* between the gathered honest parameters and (2) *different* from any extremum among the gathered honest parameters (as 1 correct parameter server was over-provisioned). Since each subset of the gathered honest parameters contains a subset of all the $n_{ps} - f_{ps}$ honest parameters, the expected maximum distance between two honest parameters is thus decreased after applying DMC.

```
Algorithm 1 Worker
 1: Get seed
 2: m ← init_model(seed)
 3: g ← compute_grad(m)
 4: $t \leftarrow 0$
 5: repeat
 6:     ms ← read_models()
 7:     m.set($Median$(ms))
 8:     g ← compute_grad(m)
 9:     $t \leftarrow t + 1$
10: until $t >$ max_steps
```

```
Algorithm 2 Server
 1: Get $T$ & seed
 2: m ← init_model(seed)
 3: $t \leftarrow 0$
 4: repeat
 5:     gs ← read_gradients()
 6:     m.update($MDA$(gs))
 7:     if $t \bmod T = 0$ then
 8:         m ← read_models()
 9:         m ← $Median$(ms)
10:     end if
11:     $t \leftarrow t + 1$
12: until $t >$ max_steps
```

Fig. 2 *ByzSGD*: worker and parameter server logic

4.2 The *ByzSGD* algorithm

Algorithms 1 and 2 (Fig. 2) depict the training loop applied by workers and servers respectively. *ByzSGD* operates iteratively in two phases: *scatter* and *gather*. One *gather* phase is entered every T steps (lines 8–11 in Algorithm 2); we call the whole T steps the *scatter* phase.

4.2.1 Initialization

As an initialization step, correct servers initialize their local model with the same random values, i.e., using the same *seed*. Concretely, each correct parameter server i and worker j starts (at step $t = 0$) with the same parameter vector:

$$\forall i \in \left[1 \ldots n_{ps} - f_{ps}\right], \theta_0^{(i)} \triangleq \theta_0$$
$$\forall j \in [1 \ldots n_w - f_w], \theta_0^{(j)} \triangleq \theta_0$$

Additionally, honest servers compute the value of T, where workers compute gradients based on the initial model.

4.2.2 Training loop

Each training step $t \in \mathbb{N}$, the following sub-steps are executed sequentially (unless otherwise stated).

1. Each parameter server i requests gradient estimations g_t from *all* workers and then applies the *MDA* function on the received gradients (whose number might not exceed $n_w - f_w$ gradients), computing the aggregated gradient $G_t^{(i)}$. Then, each server uses its own computed $G_t^{(i)}$ to update the model as follows: $\theta_{t+1/2}^{(i)} = \theta_t^{(i)} - \eta_t G_t^{(i)}$ (line 6 in Algorithm 2).
2. Each worker j then asks for the updated models from *all* servers; yet, it does not expect to receive more than $n_{ps} - f_{ps}$ replies. The worker j then computes *Median* on the received models to obtain $\theta_{t+1}^{(j)}$. In normal (i.e., *scatter*) steps, worker j sets $\theta_{t+1}^{(j)} = \theta_{t+1/2}^{(j)}$ (line 7 in

Algorithm 1). The worker then uses the updated model to compute a gradient for the new training step.

3. To bound the drifts between parameter vectors at correct servers, each $T = \frac{1}{3l\eta_1}$ steps, a global *gather* phase is entered on the server side by completing the *Distributed Median-based Contraction* module. During this phase, the following happens: each server i sends to all other servers its current view of the model $\theta_t^{(i)}$. After gathering models from $n_{ps} - f_{ps}$ servers, each server i aggregates such models with *Median*, computing $\theta_t^{(i(agg))}$. After the conclusion of the gather phase, each worker j pulls the model $\theta_t^{(i(agg))}$ from $n_{ps} - f_{ps}$ servers and aggregates the received models using *Median*. Finally, each worker j then uses the aggregated model to compute the new gradient, and the algorithm continues normally from step 1.

5 Correctness of *ByzSGD*

This section presents the proof of (α, f)-Byzantine resilience of *ByzSGD*. To do so, we first prove that for t large enough, all honest servers' parameters will be arbitrarily close to one another. Thus, their individual behavior will be well described by the behavior of the average of these parameters. We then show that the average parameter $\bar{\theta}_t$ satisfies (α, f)-Byzantine resilience, i.e.,

$$\langle \mathbb{E}\bar{G}_t, \nabla L\left(\bar{\theta}_t\right)\rangle \geq \frac{3}{4}\left\|\nabla L\left(\bar{\theta}_t\right)\right\|_2^2 \tag{7}$$

Finally, we will show that the moments of the average effective gradient $\bar{G}_t$ undergone by $\bar{\theta}_t$ are bounded, as required by the standard proof of SGD convergence [12], where the average effective gradient is defined as follows:

$$\bar{G}_t = \frac{\bar{\theta}_t - \bar{\theta}_{t+1}}{\eta_t}. \tag{8}$$

Note that for simplicity, we focus, in our correctness proof, on the case where we apply a *gather* step in each iteration, i.e., $T = 1$. Later, we show the bound on T to ensure safe convergence, and we show empirically the effect of changing T on performance in Sect. 7.

5.1 Safety of *Median*

The first key element of the proof is to prove that *Median* contracts the servers' parameters in expectation, despite the Byzantines' attacks. This turns out to be nontrivial. In fact, the diameter of servers' parameters does not decrease monotonically. The trick is to follow the evolution of another measure of the spread of the servers' parameters, namely the sum of coordinate-wise diameters. We prove that, using *Median*, Byzantines can never increase this quantity.

The *diameter* notion denotes the maximum distance between any two vectors in a set of *correct/honest* vectors. We call such a set of vectors: *family of vectors*. Interestingly, the diameters satisfy the triangle inequality, as shown by the following lemma.

Lemma 1 *The diameters and coordinate-wise diameters satisfy the triangle inequality. Namely, for any two families of vectors* $\mathbf{x}$ *and* $\mathbf{y}$*, we have the following inequality*

$$\Delta^{cw}(\mathbf{x}+\mathbf{y}) \leq \Delta^{cw}(\mathbf{x}) + \Delta^{cw}(\mathbf{y}). \tag{9}$$

As an immediate corollary, by triangle inequality of norms, for any $r \in [1, \infty]$*, we also have* $\Delta_r^{cw}(\mathbf{x}+\mathbf{y}) \leq \Delta_r^{cw}(\mathbf{x}) + \Delta_r^{cw}(\mathbf{y})$*. We also have* $\Delta_r(\mathbf{x}+\mathbf{y}) \leq \Delta_r(\mathbf{x}) + \Delta_r(\mathbf{y})$*.*

Proof For any coordinate $i \in [d]$, the following holds:

$$\Delta^{cw}(\mathbf{x}+\mathbf{y})[i] = \max_{j,k\in[h]}\left|x^{(j)}[i] + y^{(j)}[i] - x^{(k)}[i] - y^{(k)}[i]\right| \tag{10}$$

$$\leq \max_{j,k\in[h]}\left\{\left|x^{(j)}[i] - x^{(k)}[i]\right| + \left|y^{(j)}[i] - y^{(k)}[i]\right|\right\} \tag{11}$$

$$\leq \max_{j,k\in[h]}\left|x^{(j)}[i] - x^{(k)}[i]\right| + \max_{j',k'\in[h]}\left|y^{(j')}[i] - y^{(k')}[i]\right| \tag{12}$$

$$= \Delta^{cw}(\mathbf{x})[i] + \Delta^{cw}(\mathbf{y})[i], \tag{13}$$

which concludes the proof for coordinate-wise diameters. The proof for ℓ_r diameters is similar. □

Let $\Delta_{\mathbf{x}}$ the sum of coordinate-wise diameters. For instance, Δ_{θ_t} is defined as:

$$\Delta_{\theta_t} = \sum_{i=1}^{d} \max_{j,k\in[1\ldots n_{ps}-f_{ps}]}\left|\theta_t^{(j)}[i] - \theta_t^{(k)}[i]\right|. \tag{14}$$

Assuming that all delivering configurations can occur with a positive probability, we can show that there will be contraction in expectation.

Let $(d, f, n, q) \in (\mathbb{N} - \{0\})^2 \times \mathbb{N}^2$ such that $2f + 2 \leq q \leq \left\lfloor \frac{n-f}{2} \right\rfloor + f + 1$, and denote $h \triangleq n - f$.[7]

Denote $\theta_{t+1/2}^{(j)} \in \mathbb{R}^d$ the parameter vector held by parameter server j at step t (just after robustly aggregating the gradients), $\Theta_{t+1/2}$ the parameter vectors from correct parameter servers that were delivered by parameter server j at step t, and $z_t^{(j)} = (z_t^{(1,j)}, \ldots, z_t^{(f,j)})$ the Byzantine parameter vectors[8] that were delivered by parameter server j at step t, where each $z_t^{(k,j)} \in \mathbb{R}^d$.

[7] This whole subsection is about the behavior of the parameter servers hence, we remove the subscript ps from all variables, e.g., $n \triangleq n_{ps}$.

[8] A Byzantine parameter server can send different parameter vectors to each correct parameter server at each step t, hence the notation.

The updated vector of parameter server j at step t is then given by

$$\theta_{t+1}^{(j)} \triangleq Median\left(\theta_{t+1/2}^{(j)}, \Theta_{t+1/2}, z_t^{(j)}\right). \tag{15}$$

In the following, we prove that the updated parameters $\theta_{t+1}^{(1)} \dots \theta_{t+1}^{(h)}$ have been contracted compared to parameters $\theta_{t+1/2}^{(1)} \dots \theta_{t+1/2}^{(h)}$. It turns out that this claim does not actually hold for naive measures of how "contracted" a set of parameters are. In particular, the diameter of the vectors $\theta_{t+1}^{(1)} \dots \theta_{t+1}^{(h)}$'s measured in, say, ℓ_2 norm, does not necessarily get contracted.

The key trick of our analysis is to focus instead on coordinate-wise diameters. More precisely, denote:

$$\varDelta_{\theta_{t+1/2},i} \triangleq \max_{j,k\in[1\dots h]} \left|\theta_{t+1/2}^{(j)}[i] - \theta_{t+1/2}^{(k)}[i]\right|$$

the diameter of vectors $\theta_{t+1/2}^{(1)} \dots \theta_{t+1/2}^{(h)}$ along coordinate i. We can then add up these coordinate-wise diameters to obtain the measure $\varDelta_{\theta_{t+1/2}} = \sum_{i\in[1\dots d]} \varDelta_{\theta_{t+1/2},i}$ which is still a measure of how spread the vectors $\theta_{t+1/2}^{(j)}$'s are. One key result of this subsection is that this measure decreases exponentially, i.e.,

$$\mathbb{E}[\varDelta_{\theta_{t+1}}] \leq m\varDelta_{\theta_{t+1/2}}, \tag{16}$$

for some contraction parameter $m \in [0, 1[$, and where the expectation is taken over $X \sim \mathcal{S}$. More precisely, in expectation over random delivering configurations, no matter how Byzantines attack the different workers, the spread of the parameter servers parameters decrease exponentially.

The following lemma shows that bounding this measure is sufficient to control the diameter of servers' parameters.

Lemma 2 *We have the following inequalities:*

$$\begin{aligned}\max_{j,k\in[1\dots h]} &\left\|\theta_{t+1/2}^{(j)} - \theta_{t+1/2}^{(k)}\right\|_2 \\ &\leq \max_{j,k\in[1\dots h]} \left\|\theta_{t+1/2}^{(j)} - \theta_{t+1/2}^{(k)}\right\|_1 \leq \varDelta_{\theta_{t+1/2}} \\ &\leq d \max_{j,k\in[1\dots h]} \left\|\theta_{t+1/2}^{(j)} - \theta_{t+1/2}^{(k)}\right\|_1 \\ &\leq d^{3/2} \max_{j,k\in[1\dots h]} \left\|\theta_{t+1/2}^{(j)} - \theta_{t+1/2}^{(k)}\right\|_2.\end{aligned}$$

Proof Let j^* and k^* two servers whose parameters are most distant, i.e., such that

$$\max_{j,k\in[1\dots h]} \left\|\theta_{t+1/2}^{(j)} - \theta_{t+1/2}^{(k)}\right\|_1 = \sum_{i=1}^{d} \left|\theta_{t+1/2}^{(j^*)}[i] - \theta_{t+1/2}^{(k^*)}[i]\right|.$$

Clearly, the latter term is upper-bounded by

$$\sum_{i=1}^{d} \max_{j,k\in[1\dots h]} \left|\theta_{t+1/2}^{(j)}[i] - \theta_{t+1/2}^{(k)}[i]\right|,$$

which is exactly the sum of coordinate-wise diameters ($\varDelta_{\theta_{t+1/2}}$). The first inequality is derived from the well-known equality $\|z\|_2 \leq \|z\|_1$.

Note that a coordinate-wise distance is smaller than the ℓ_1 distance. Therefore,

$$\begin{aligned}\varDelta_{\theta_{t+1/2}} &= \sum_{i=1}^{d} \max_{j,k} \left|\theta_{t+1/2}^{(j)}[i] - \theta_{t+1/2}^{(k)}[i]\right| \\ &\leq \sum_{i=1}^{d} \max_{j,k} \left\|\theta_{t+1/2}^{(j)} - \theta_{t+1/2}^{(k)}\right\|_1 \\ &= d \cdot \max_{j,k} \left\|\theta_{t+1/2}^{(j)} - \theta_{t+1/2}^{(k)}\right\|_1.\end{aligned}$$

We then conclude by using the inequality $\|z\|_1 \leq \sqrt{d}\,\|z\|_2$. □

Let $x_{min}[i]$ and $x_{max}[i]$ the minimum and maximum i-coordinate of correct vectors $\theta_{t+1/2}^{(1)} \dots \theta_{t+1/2}^{(h)}$'s.

Lemma 3 $\varDelta_{\theta_{t+1},i} \leq \varDelta_{\theta_{t+1/2},i}$ *with probability 1. In other words, there can be no dilation of the diameter along a coordinate.*

Proof Since $q \geq 2f + 2 > 2f$, we know that a strict majority of vectors delivered to j are correct, and thus belong to $[x_{min}[i], x_{max}[i]]$. Therefore, $\theta_{t+1}^{(j)}[i]$ must belong to this interval too. Since this holds for any parameter server j, we know that $\varDelta_{\theta_{t+1},i} \leq x_{max}[i] - x_{min}[i] = \varDelta_{\theta_{t+1/2},i}$. □

Now along each coordinate, we separate the set of workers along two subsets $left[i]$ and $right[i]$, defined by

$$\begin{aligned}left[i] &= \left\{j \in [1\dots h] \;\middle|\; \theta_{t+1/2}^{(j)}[i] \leq \frac{1}{2}(x_{min}[i] + x_{max}[i])\right\}, \\ right[i] &= [1\dots h] - left[i]\end{aligned}$$

We then go on showing that the minority subset has a positive probability of making a significant step towards the other side. First, we quantify the size of this large possible step.

Lemma 4 *If* $X_j \subset right[i]$*, then*

$$\theta_{t+1}^{(j)}[i] \geq \frac{1}{4}(3\,x_{min}[i] + x_{max}[i]).$$

Moreover, if $X_j \subset left[i]$*, then*

$$\theta_{t+1}^{(j)}[i] \leq \frac{1}{4}(x_{min}[i] + 3\,x_{max}[i]).$$

Proof Assume $X_j \subset right[i]$. Since $q \geq 2f+2$, we know that $\left|j \cup X_j\right| \geq q - f \geq q/2 + 1$ contains a strict majority of the inputs to compute $\theta_{t+1}^{(j)}[i]$. Thus the median is at least the average of two smallest coordinates i of points $\left|j \cup X_j\right|$, which is at least

$$\theta_{t+1}^{(j)}[i] \geq \frac{\theta_{t+1/2}^{(j)}[i] + \frac{1}{2}\left(x_{min}[i] + x_{max}[i]\right)}{2} \geq \frac{1}{4}\left(3\,x_{min}[i] + x_{max}[i]\right)$$

The second bound is proved similarly. □

Lemma 5 *We can then show expected strict contraction along every coordinate i, i.e.,*

$$\mathbb{E}\left[\Delta_{\theta_{t+1},i}\right] \leq \left(1 - \frac{\rho}{4}\right)\Delta_{\theta_{t+1/2},i}.$$

Proof Note that $right[i] + left[i] = h$. Thus at least one of the two subsets has at most $\lfloor h/2 \rfloor$ elements. Since $q \leq \lfloor \frac{n-f}{2} \rfloor + f + 1 = \lfloor h/2 \rfloor + f + 1$, we know that $q - f \leq \lfloor h/2 \rfloor + 1$. This means that there is a subset s_0 of $left[i]$ or of $right[i]$ with at least $q - f$ elements.

Without loss of generality, assume $s_0 \subset left[i]$ with cardinal $q - f$, and let $s_j = s_0 - \{j\}$. Then the tuple $s = (s_1, \ldots, s_h) \in S$ is a delivering configuration. By virtue of our key assumption (Equation 6), we know that $P(X = s) \geq \rho > 0$.

But then, by the previous lemma, in the event $X = s$, we have $\theta_{t+1}^{(j)}[i] \geq \frac{1}{4}\left(3\,x_{min}[i] + x_{max}[i]\right)$ for all servers j. Since, moreover, we can easily verify that $\theta_{t+1}^{(j)}[i] \leq x_{max}[i]$, we conclude that

$$\Delta_{\theta_{t+1},i} \leq x_{max}[i] - \frac{1}{4}\left(3\,x_{min}[i] + x_{max}[i]\right) = \frac{3}{4}\left(x_{max}[i] - x_{min}[i]\right) = \frac{3\Delta_{\theta_{t+1/2},i}}{4}.$$

The same bound can be derived for the case $s_0 \subset right[i]$.

Now note that in the event $X \neq s$, by Lemma 3, we still have $\Delta_{\theta_{t+1},i} \leq \Delta_{\theta_{t+1/2},i}$. We can now take the average of the two above cases, which yields

$$\begin{aligned}\mathbb{E}[\Delta_{\theta_{t+1},i}] &= \mathbb{E}\left[\Delta_{\theta_{t+1},i} \,\middle|\, X = s\right] P(X = s) \\ &\quad + \mathbb{E}\left[\Delta_{\theta_{t+1},i} \,\middle|\, X \neq s\right](1 - P(X = s)) \\ &= \mathbb{E}\left[\Delta_{\theta_{t+1},i} \,\middle|\, X \neq s\right] \\ &\quad + \left(\mathbb{E}\left[\Delta_{\theta_{t+1},i} \,\middle|\, X = s\right] - \mathbb{E}\left[\Delta_{\theta_{t+1},i} \,\middle|\, X \neq s\right]\right) P(X = s) \\ &\leq \Delta_{\theta_{t+1/2},i} + \left(\frac{3\Delta_{\theta_{t+1/2},i}}{4} - \Delta_{\theta_{t+1/2},i}\right)\rho \\ &= \left(1 - \frac{\rho}{4}\right)\Delta_{\theta_{t+1/2},i},\end{aligned}$$

which concludes the proof. □

Lemma 6 *Despite any Byzantine attack $z(X)$ that reacts to the random choice of delivering configuration $X \sim \mathcal{S}$, there is a strict contraction of the sum of coordinate-wise diameters, i.e.,*

$$\forall z,\ \ \mathbb{E}_{X\sim S}\left[\Delta_{\theta_{t+1}}\right] \leq m\Delta_{\theta_{t+1/2}},$$

where $m < 1$ only depends on the probability distribution $\mathcal{S}$.

Proof We simply use the linearity of the expectation, which yields

$$\mathbb{E}[\Delta_y] = \sum_{i=1}^{d}\mathbb{E}[\Delta_{\theta_{t+1},i}] \leq \sum_{i=1}^{d}\left(1 - \frac{\rho}{4}\right)\Delta_{\theta_{t+1/2},i} = \left(1 - \frac{\rho}{4}\right)\Delta_{\theta_{t+1/2}},$$

which is a key lemma in our proof. □

Unfortunately, this is still not sufficient to guarantee the contraction of the servers' parameters, because of a potential drift during learning. However, as the learning rate decreases, we show that contraction eventually becomes inevitable.

In the following, we show that the drift among honest parameter servers is bounded. Concretely, there exists constants A and B such that $\mathbb{E}\left[\Delta_{\theta_{t+1}}\right] \leq (m + A\eta_t)\Delta_{\theta_t} + B\eta_t$. Here, the expectation is over delivering configurations, from parameter servers to workers, from workers to parameter servers and in-between parameter servers. The expectation is also taken over stochastic gradient estimates by workers, and for any attacks by the Byzantines.

5.2 Eventual contraction of parameters

Consider parameters θ_t at time t. Notice that this parameter will undergo three operations, each of which will be attacked by Byzantines.

First, each worker j will be delivered a subset of parameters. It will need to compute the median of the parameters, to obtain a model $\theta_t^{(j)}$. It will then compute an estimate $g_t^{(j)}$ of the gradient $\nabla L\left(\theta_t^{(j)}\right)$, which will then be broadcast to parameter servers. We make the following critical observations.

Lemma 7 *For any two workers j and k, the noises on gradient estimates is bounded*

$$\mathbb{E} \max_{j,k \in [1 \ldots h_w]} \left\| \xi_t^{(j)} - \xi_t^{(k)} \right\|_2 \leq 2\sigma' \sqrt{h_w}.$$

Proof Since $a \mapsto \sqrt{a}$ is a concave function, Jensen's inequality asserts that $\mathbb{E}\sqrt{X} \leq \sqrt{\mathbb{E}X}$. Using in addition the inequality $(a-b)^2 \leq 2a^2 + 2b^2$, we obtain

$$\mathbb{E} \max_{j,k \in [1 \ldots h_w]} \left\| \xi_t^{(j)} - \xi_t^{(k)} \right\|_2 = \mathbb{E} \sqrt{\max_{j,k \in [h_w]} \left\| \xi_t^{(j)} - \xi_t^{(k)} \right\|_2^2} \tag{17}$$

$$\leq \sqrt{2\mathbb{E} \max_{j,k \in [h_w]} \left\{ \left\| \xi_t^{(j)} \right\|_2^2 + \left\| \xi_t^{(k)} \right\|_2^2 \right\}} \tag{18}$$

$$= 2\sqrt{\mathbb{E} \max_{j \in [h_w]} \left\| \xi_t^{(j)} \right\|_2^2}. \tag{19}$$

We now use the fact that the maximum over nodes $j \in [h_w]$ is smaller than the sum over nodes $j \in [h_w]$, yielding

$$\mathbb{E} \max_{j,k \in [1 \ldots h_w]} \left\| \xi_t^{(j)} - \xi_t^{(k)} \right\|_2$$

$$\leq 2\sqrt{\mathbb{E} \sum_{j \in [h_w]} \left\| \xi_t^{(j)} \right\|_2^2} \tag{20}$$

$$= 2\sqrt{\sum_{j \in [h_w]} \mathbb{E} \left\| \xi_t^{(j)} \right\|_2^2} \tag{21}$$

$$\leq 2\sqrt{\sum_{j \in [h_w]} \sigma'^2} = 2\sigma' \sqrt{h_w}, \tag{22}$$

where the last inequality uses the assumption on bounded variance among gradient estimations. □

Lemma 8 *For any two workers j and k, the delivered gradients satisfy*

$$\mathbb{E} \max_{j,k \in [1 \ldots h_w]} \left\| g_t^{(j)} - g_t^{(k)} \right\|_2 \leq 2h_w\sigma' + l\Delta_{\theta_t},$$

where the expectation is over the random delivering configuration of parameters from servers to workers and over the random estimation of the gradients.

Proof Note that

$$\left\| g_t^{(j)} - g_t^{(k)} \right\|_2 \leq \left\| g_t^{(j)} - \nabla L\left(\theta_t^{(j)}\right) \right\|_2 + \left\| \nabla L\left(\theta_t^{(j)}\right) - \nabla L\left(\theta_t^{(k)}\right) \right\|_2 + \left\| \nabla L\left(\theta_t^{(k)}\right) - g_t^{(k)} \right\|_2.$$

As a result, we know that

$$\max_{j,k \in [1 \ldots h_w]} \left\| g_t^{(j)} - g_t^{(k)} \right\|_2 \leq 2 \max_{j \in [1 \ldots h_w]} \left\| g_t^{(j)} - \nabla L\left(\theta_t^{(j)}\right) \right\|_2 + \max_{j,k \in [1 \ldots h_w]} \left\| \nabla L\left(\theta_t^{(j)}\right) - \nabla L\left(\theta_t^{(k)}\right) \right\|_2. \tag{23}$$

We now use the fact that

$$\mathbb{E} \max_{j \in [1 \ldots h_w]} \left\| g_t^{(j)} - \nabla L\left(\theta_t^{(j)}\right) \right\|_2 \leq \mathbb{E} \sum_{j=1}^{h_w} \left\| g_t^{(j)} - \nabla L\left(\theta_t^{(j)}\right) \right\|_2 = \sum_{j=1}^{h_w} \mathbb{E} \left\| g_t^{(j)} - \nabla L\left(\theta_t^{(j)}\right) \right\|_2.$$

Using our assumption of unbiased gradient estimator with uniformly bounded error σ', we see that

$$\mathbb{E} \max_{j \in [1 \ldots h_w]} \left\| g_t^{(j)} - \nabla L\left(\theta_t^{(j)}\right) \right\|_2 \leq h_w\sigma'.$$

To bound the second term of Eq. (23), we now use the fact that $\left\| \theta_t^{(j)} - \theta_t^{(k)} \right\|_2 \leq \Delta_{\theta_{t+1/2}} \leq \Delta_{\theta_t}$. Since ∇L is l-Lipschitz, we thus have

$$\left\| \nabla L\left(\theta_t^{(j)}\right) - \nabla L\left(\theta_t^{(k)}\right) \right\|_2 \leq l\Delta_{\theta_t}.$$

This equation holds in particular for the maximum of the left-hand side, as we vary j and k. This concludes the proof of the lemma. □

We now move on to the guarantee with respective to the second attack of Byzantines, which occurs as servers aggregate workers' estimations of the gradient. This guarantee is provided by the Byzantine resilience of *MDA*.

First, we note that *MDA* ensures that its output gradient lies within the correct set of gradients submitted to a correct server, as stated by the following lemma.

Lemma 9 *The diameter of the aggregations of gradients by* MDA *is at most three times the diameter of correct gradients. Denote with $G_t^{(j)}$ the aggregated gradient at server j, we have:*

$$\max_{j,k \in [1 \ldots h_{ps}]} \left\| G_t^{(j)} - G_t^{(k)} \right\|_2 \leq 3 \max_{r,s \in [1 \ldots h_w]} \left\| g_t^{(r)} - g_t^{(s)} \right\|_2. \tag{24}$$

Proof Let us first focus on the computation of $G_t^{(j)}$. Recall that it is obtained by gathering q_w vectors $g_t^{(r)}$ from workers, including at most f_w Byzantines, and computing the *MDA* of the the collected vectors. Recall that *MDA* averages the subset of vectors of minimal diameter. Let $\mathcal{X}^*$ a subset of gradients of size $q_w - f_w$ that minimizes the diameter of the gradients.

Note that the diameter of delivered correct gradients is necessarily at most the diameter of all correct gradients, which we shall denote

$$D(g_t) = \max_{j,k\in[1\dots h_w]} \left\| g_t^{(j)} - g_t^{(k)} \right\|_2 . \tag{25}$$

Since $q_w \geq 2f_w + 1$, there is a subset of size $q_w - f_w$ that only contains correct gradients, and whose diameter is thus at most $D(g_t)$.

Therefore the subset $\mathcal{X}^*$ must have diameter at most $D(g_t)$. But since $q_w \geq 2f_w + 1$, we know that at least one correct gradient $g_t^{(r^*(j))}$ belongs to this subset. This means that all gradients collected by $\mathcal{X}^*$ must be at distance at most $D(g_t)$ from $g_t^{(r^*(j))}$. In other words, for any $G_t^{(j)}$, there exists a correct gradient $g_t^{(r^*(j))}$ such that $\left\| G_t^{(j)} - g_t^{(r^*(j))} \right\|_2 \leq D(g_t)$.

But then, we have

$$\begin{aligned}\left\| G_t^{(j)} - G_t^{(k)} \right\|_2 \leq & \left\| G_t^{(j)} - g_t^{(r^*(j))} \right\|_2 \\ & + \left\| g_t^{(r^*(j))} - g_t^{(r^*(k))} \right\|_2 \\ & + \left\| g_t^{(r^*(k))} - G_t^{(k)} \right\|_2 ,\end{aligned}$$

which is at most $3D(g_t)$. □

Recall that each server j's parameters are now updated by adding $-\eta_t G_t^{(j)}$ to $\theta_t^{(j)}$. We can bound the drift that this update causes as follows.

Lemma 10 *We have the following inequality:*

$$\mathbb{E}\left[\Delta_{\theta_t - \eta_t G_t}\right] \leq 6d\eta_t h_w \sigma' + (1 + 3d\eta_t l)\Delta_{\theta_t},$$

where the average is taken over delivering configurations and stochastic gradient estimates.

Proof Note that, on each coordinate i, for any two servers j and k, we have

$$\begin{aligned} & \left| \left(\theta_t^{(j)}[i] - \eta_t G_t^{(j)}[i]\right) - \left(\theta_t^{(k)}[i] - \eta_t G_t^{(k)}[i]\right) \right| \\ \leq & \left| \theta_t^{(j)}[i] - \theta_t^{(k)}[i] \right| + \eta_t \left| G_t^{(j)}[i] - G_t^{(k)}[i] \right| \\ \leq & \left| \theta_t^{(j)}[i] - \theta_t^{(k)}[i] \right| + \eta_t \left\| G_t^{(j)} - G_t^{(k)} \right\|_2 \\ \leq & \ \Delta_{\theta_t}[i] + \eta_t \max_{j',k'\in[1\dots h_{ps}]} \left\| G_t^{(j')} - G_t^{(k')} \right\|_2 . \end{aligned}$$

Note that the right-hand side is now independent from j and k, and is thus unchanged as we take the maximum over all j and k's. Summing over all coordinates i yields

$$\Delta_{\theta_t - \eta_t G_t} \leq \Delta_{\theta_t} + d\eta_t \max_{j',k'\in[1\dots h_{ps}]} \left\| G_t^{(j')} - G_t^{(k')} \right\|_2 .$$

We now take the average, and invoke the two previous lemmas to derive the result. □

Finally, we can combine the result with the contraction property of the *Median*.

Lemma 11 *We have the following inequality:*

$$\mathbb{E}\left[\Delta_{\theta_{t+1}}\right] \leq \left(1 + 3d\eta_t l - \frac{\rho}{4}\right)\Delta_{\theta_t} + 6d\eta_t h_w \sigma'.$$

Proof This is an immediate application of Lemmas 5 and 10, and using $(1+a)(1-b) \leq 1 + a - b$ for $a, b \geq 0$. □

We are now almost there. We will need this elementary lemma to conclude.

Lemma 12 *Let* $k \in [0, 1[$ *and* $\delta_t > 0$ *be a positive decreasing sequence such that* $\delta_t \to 0$. *Then there exists constants* $C > 0$, *which depend on* k *and* δ_0, *such that*

$$\sum_{i=1}^{t} k^{t-i}\delta_i \leq Ck^{t/2} + \frac{\delta_{\lfloor t/2 \rfloor}}{1-k}. \tag{26}$$

In particular, the left-hand side converges to zero.

Proof We divide the sum between elements before than $s = \lfloor t/2 \rfloor$ and elements after this threshold. This yields:

$$\begin{aligned}\sum_{i=1}^{t} k^{t-i}\delta_i &= \sum_{i=1}^{s-1} k^{t-i}\delta_i + \sum_{i=1+s}^{t} k^{t-i}\delta_i \\ &\leq k^{1+t-s}\delta_0 \sum_{j=1}^{s-1} k^j + \delta_s \sum_{i=0}^{t-s} k^j \\ &\leq \frac{\delta_0 k^{1+t-s}}{1-k} + \frac{\delta_s}{1-k}.\end{aligned}$$

Since $1 + t - s \leq 1 + t - (t/2 - 1) = t/2$, defining $C = \delta_0/(1-k)$ allows to conclude. □

Finally, we can derive the gathering of *ByzSGD*.

Lemma 13 (Gathering of *ByzSGD*) *If learning rates go to zero* ($\eta_t \to 0$), *then servers converge to the same state, and their diameter is of the order of the learning rate. More precisely, there exists a constant* $C > 0$ *and a time* t_0 *such that, for* $t \geq t_0$,

$$\mathbb{E} \max_{j,k\in[1\dots h_{ps}]} \left\| \theta_t^{(j)} - \theta_t^{(k)} \right\|_2 \leq C\left(1 - \frac{\rho}{8}\right)^{t/2} + \frac{24dl\eta_{\lfloor t/2 \rfloor}}{\rho}.$$

Proof Assume $0 \leq u_{t+1} \leq (1 + A\delta_t - 2\varepsilon)u_t + \delta_t$, where $\varepsilon = \rho/8$, $\delta_t = 3d\eta_t l$ and $A = \frac{l}{2h_w\sigma^7}$. Since $\delta_t \to 0$, we know that there is a time t_0 such that for $t \geq t_0$, we have $C\delta_t \leq \varepsilon$. For $t \geq t_0$, we then have $u_{t+1} \leq ku_t + \delta_t$, where $k \leq 1 - \rho/8 < 1$ and $\delta_t \to 0$. But then, we observe that, for $s \geq 0$,

$$\begin{aligned} u_{s+t_0} &\leq k^s u_{t_0} + \sum_{i=1}^{s} k^{i-1}\delta_{t_0+s-i} \\ &\leq k^s u_{t_0} + \sum_{j=1}^{s} k^{s-j}\delta_{t_0-1+j}. \end{aligned}$$

Using Eq. (26), we conclude that, for $t = s+t_0 \geq t_0$, we have $u_t \leq Ck^{(t-t_0)/2} + \frac{\delta_{t_0-1+\lfloor s/2\rfloor}}{1-k}$. Applying this to $u_t = \mathbb{E}[\Delta_{\theta_t}]$, redefining the constant C and noting that $t_0 - 1 + \lfloor s/2\rfloor \geq \lfloor (s+t_0)/2\rfloor$, implies the bound of the lemma.

We can finally conclude by noting that

$$\max_{j,k\in[1\ldots h_{ps}]} \left\|\theta_t^{(j)} - \theta_t^{(k)}\right\|_2 \leq \Delta_{\theta_t}.$$

□

In particular, assuming, say $\eta_t = 1/t^\alpha$, the expected diameter is of the order $\mathcal{O}\left(\frac{dl}{\rho t^\alpha}\right)$.

Theorem 1 (Eventual contraction of Median) *As a corollary, assuming $\eta_t \to 0$, we have $\mathbb{E}[\Delta_{\theta_t}] \to 0$.*

Proof This comes from the eventual contraction of Δ_{θ_t}. Note that it implies that some other measures of the spread of servers' parameters, like their diameter measured in ℓ_2, also converge to zero. □

5.2.1 Estimating the gathering frequency

Given Lemma 9, we can show that the distance between aggregated gradients on two correct parameter servers, at any time t, is bounded. Hence, SGD, with *MDA*, alone would converge.

Now, to satisfy the required assumption by *MDA* (Eq. 3, Sect. 2.4), models at correct parameter servers should not go arbitrarily far from each other. Thus, a global *gather* phase (step 3 in the *ByzSGD* algorithm) is executed once in a while to bring the correct models back close to each other. Yet, the question is: how many iterations per scatter step can be executed without breaking such an assumption? We quantify the maximum number of steps that can be executed in one *scatter* phase before executing one *gather* phase. Hence, the goal is to find the maximum possible distance between correct models that still satisfies the requirement of *MDA* on the distance between correct gradients (Eq. 3).

Without loss of generality, assume two correct parameter servers x and z starting with the same initial model θ_0. After the first step, their updated models are given by:

$$\begin{aligned} \theta_1^{(x)} &= \theta_0 - \eta_1 MDA\left(g_1^{(1)}\ldots g_1^{(n_w)}\right)_x \\ \theta_1^{(z)} &= \theta_0 - \eta_1 MDA\left(g_1^{(1)}\ldots g_1^{(n_w)}\right)_z \end{aligned}$$

Thus, the difference between them is given by:

$$\begin{aligned} &\left\|\theta_1^{(x)} - \theta_1^{(z)}\right\|_2 \\ &= \eta_1 \left\|MDA\left(g_1^{(1)}\ldots g_1^{(n_w)}\right)_z - MDA\left(g_1^{(1)}\ldots g_1^{(n_w)}\right)_x\right\|_2 \end{aligned}$$

In a perfect environment, with no Byzantine workers, this difference is zero, since the input gradients to the *MDA* function at both servers are the same (no worker lies about its gradient estimation, i.e., there is no equivocation), and the *MDA* function is deterministic (i.e., the output of *MDA* computation on both servers is the same). However, a Byzantine worker can send different gradients to different servers while crafting these gradients carefully to trick the *MDA* function to include them in the aggregated gradient (i.e., force *MDA* to select the malicious gradients in the set $\mathcal{S}$). In this case, $\left\|\theta_1^{(x)} - \theta_1^{(z)}\right\|_2$ is not guaranteed to be zero. Based on Eq. (24), the difference between the result of applying *MDA* in the same step is bounded and hence, such a difference can be given by:

$$\left\|\theta_1^{(x)} - \theta_1^{(z)}\right\|_2 \leq 3 \cdot \eta_1 \cdot \max_{(i,j)\in H} \left\|g_1^{(i)} - g_1^{(j)}\right\|_2 \quad (27)$$

Following the same analysis, the updated models in the second step at our subject parameter servers are given by:

$$\begin{aligned} \theta_2^{(x)} &= \theta_1^{(x)} - \eta_2 MDA\left(g_2^{(1)}\ldots g_2^{(n_w)}\right)_x \\ \theta_2^{(z)} &= \theta_1^{(z)} - \eta_2 MDA\left(g_2^{(1)}\ldots g_2^{(n_w)}\right)_z \end{aligned}$$

Thus, the difference between models now will be:

$$\begin{aligned} \left\|\theta_2^{(x)} - \theta_2^{(z)}\right\|_2 = &\left\| \left(\theta_1^{(x)} - \eta_2 MDA\left(g_2^{(1)}\ldots g_2^{(n_w)}\right)_x\right) \right. \\ &\left. - \left(\theta_1^{(z)} - \eta_2 MDA\left(g_2^{(1)}\ldots g_2^{(n_w)}\right)_z\right) \right\| \\ \leq &\left\|\theta_1^{(x)} - \theta_1^{(z)}\right\|_2 \\ &+ \eta_2 \left\| MDA\left(g_2^{(1)}\ldots g_2^{(n_w)}\right)_x \right. \\ &\left. - MDA\left(g_2^{(1)}\ldots g_2^{(n_w)}\right)_z \right\| \end{aligned}$$

The bound on the first term is given in Eq. (27) and that on the second term is given in Eq. (24) and hence, the difference between models in the second step is given by:

$$\left\|\theta_2^{(x)} - \theta_2^{(z)}\right\|_2 \leq 3 \cdot \eta_1 \cdot \max_{(i,j)\in H} \left\|g_1^{(i)} - g_1^{(j)}\right\|_2 + 3 \cdot \eta_2 \cdot \max_{(i,j)\in H} \left\|g_2^{(i)} - g_2^{(j)}\right\|_2 \tag{28}$$

By induction, we can write that the difference between models on two correct parameter servers at step τ is given by:

$$\left\|\theta_t^{(x)} - \theta_t^{(z)}\right\|_2 \leq \sum_{t=1}^{\tau} 3 \cdot \eta_t \cdot \max_{(i,j)\in H} \left\|g_t^{(i)} - g_t^{(j)}\right\|_2 \tag{29}$$

Since $g_t^{(i)}$ and $g_t^{(j)}$ are computed at different workers, they can be computed based on different models $\theta_t^{(i)}$ and $\theta_t^{(j)}$. Following the Lipschitzness of the loss function,

$$\left\|g_t^{(i)} - g_t^{(j)}\right\|_2 \lesssim l \left\|\theta_t^{(i)} - \theta_t^{(j)}\right\|_2 .$$

Noting that the sequence η_t is monotonically decreasing with $t \to \infty$, Eq. (29) can be written as:

$$\left\|\theta_t^{(x)} - \theta_t^{(z)}\right\|_2 \lesssim 3 \cdot \eta_1 \cdot l \sum_{t=1}^{\tau} \max_{(i,j)\in H} \left\|\theta_t^{(i)} - \theta_t^{(j)}\right\|_2$$

Assuming that the maximum difference between any two correct models is bounded by $\mathcal{K}$ (this is enforced anyway by the algorithm through entering frequently the *gather* phase), this difference can be written as:

$$\left\|\theta_t^{(x)} - \theta_t^{(z)}\right\|_2 \lesssim 3 \cdot \eta_1 \cdot l \cdot \mathcal{K} \cdot \tau$$

Now, to ensure the bound on the maximum difference between models, we need the value of $\left\|\theta_t^{(x)} - \theta_t^{(z)}\right\|_2 \leq \mathcal{K}$. At this point, the number of steps $\tau = T$ should be bounded from above as follows:

$$T \lesssim \frac{1}{3 \cdot \eta_1 \cdot l} \tag{30}$$

T here represents the maximum number of steps that are allowed to happen in the *scatter* phase, i.e., before entering one *gather* phase. Doing more steps than this number leads to breaking the requirement of *MDA* on the variance between input gradients, leading to breaking its Byzantine resilience guarantees. Thus, this bound is a *safety* bound that one should not pass to guarantee contraction. One can do less number of steps (than T) during the *scatter* phase for a better performance (as we discuss in Sect. 8). Moreover, this bound requires that the initial setup satisfies the assumptions of *MDA*. Having a deployment that does not follow such assumptions leads to breaking guarantees of our protocol (as we show in Sect. 7).

5.3 Liveness of server parameters

The previous section showed that eventually, the parameters will all have nearly identical values. Let us now use the results of the previous lemmas to show that, while gathering, the trajectory of parameters $\theta_t^{(j)}$ is nearly a stochastic gradient descent. The trick to do so will be to determine that the actual update of the parameters after contraction satisfies the (α, f)-Byzantine resilience condition of [11]. This actual update is what we call the the *effective gradient*.

Definition 2 The (stochastic) effective gradient $\hat{G}_t^{(j)}$ of node j is defined by

$$\hat{G}_t^{(j)} = \frac{\theta_t^{(j)} - \theta_{t+1}^{(j)}}{\eta_t}. \tag{31}$$

In particular, we shall focus on the effective gradient of the average parameter, which turns out to also be the average effective gradient, that is,

$$\bar{G}_t \triangleq \frac{1}{h_{ps}} \sum_{j\in[h_{ps}]} \hat{G}_t^{(j)} = \frac{\bar{\theta}_t - \bar{\theta}_{t+1}}{\eta_t}. \tag{32}$$

The fundamental property of the average effective gradient is to be pointing, under our assumptions, strictly in the same direction as the true gradient, for large enough values of t, i.e.,

$$\exists \alpha > 0,\ \mathbb{E}\bar{G} \cdot \nabla L\left(\bar{\theta}_t\right) \geq (1 - \sin^2 \alpha) \left\|\nabla L\left(\bar{\theta}_t\right)\right\|_2^2 .$$

To prove this for $\alpha = \pi/6$, we first show that for t large enough, all servers receive roughly the same gradients, using the previous lemmas.

Lemma 14 *The following bound holds:*

$$\mathbb{E} \max_{j\in[1\ldots h_{ps}]} \left\|G_t^{(j)} - \nabla L\left(\bar{\theta}_t\right)\right\|_2 \leq 3h_w\sigma' + 3l\Delta_{\theta_t} .$$

Proof Note that, for any server j and worker k, we have

$$\left\|G_t^{(j)} - \nabla L\left(\bar{\theta}_t\right)\right\|_2 \leq \left\|G_t^{(j)} - g_t^{(k)}\right\|_2 + \left\|g_t^{(k)} - \nabla L\left(\theta_t^{(k)}\right)\right\|_2 + \left\|\nabla L\left(\theta_t^{(k)}\right) - \nabla L\left(\bar{\theta}_t\right)\right\|_2 . \tag{33}$$

Because we use *MDA*, by virtue of Lemma 9, we know that the first term is upper bounded by $2h_w\sigma' + l\Delta_{\theta_t}$. The second term is bounded by the assumption on the size of the noise, while the last term is bounded using the Lipschitz continuity of the gradient, and the fact that the maximal distance between two models is upper-bounded by Δ_{θ_t}. Combining it all yields the lemma. □

Lemma 15 *The following bound holds:*

$$\mathbb{E} \max_{j\in[1\ldots h_{ps}]} \left\| G_t^{(j)} - \nabla L\left(\bar{\theta}_t\right) \right\|_2 \leq 3h_w\sigma' + \tag{34}$$

$$3l\left(C\left(1-\frac{\rho}{8}\right)^{t/2} + \frac{24dl\eta_{\lfloor t/2\rfloor}}{\rho}\right). \tag{35}$$

Proof Combing Lemma 13 with Lemma 14 yields the result. □

Lemma 16 *The diameters of the servers' updates after applying gradients is upper-bounded as follows:*

$$\max_{j,k\in[1\ldots h_{ps}]} \left\| \theta_t^{(j)} - \eta_t G_t^{(j)} - \left(\theta_t^{(k)} - \eta_t G_t^{(k)}\right) \right\|_2$$
$$\leq 6\eta_t h_w \sigma' + (3l+1)\Delta_{\theta_t}.$$

Proof This is derived from the triangle inequality and from previous lemmas. □

Lemma 17 *Assuming $\eta_t = 1/t^\alpha$, after applying gradients and after contraction, the distance between the overall motion of parameters of server j and the true gradient is upper-bounded as follows:*

$$\mathbb{E}\bar{G}_t \cdot \nabla L\left(\bar{\theta}_t\right) \geq \left\|\nabla L\left(\bar{\theta}_t\right)\right\|_2^2 - \left(9h_w\sigma' + \frac{Cdl^2}{\rho}\right)\left\|\nabla L\left(\bar{\theta}_t\right)\right\|_2,$$

where C is a constant.

Proof Note that $\theta_{t+1}^{(j)}$ is obtained by taking *Median* over updated parameters $\theta_t^{(k)} - \eta_t G_t^{(k)}$. By the guarantee of *Median*, we know that the ℓ_2 diameter of the (attacked) outputs is at most the diameter of the input. This implies that

$$\left\| \theta_{t+1}^{(j)} - \left(\theta_t^{(j)} - \eta_t G_t^{(j)}\right) \right\|_2 \leq \Delta_{\theta_t - \eta_t G_t}.$$

From this, and using the previous lemmas, we derive the fact that

$$\mathbb{E}\left(\theta_{t+1}^{(j)} - \theta_t^{(j)}\right) \cdot \nabla L\left(\bar{\theta}_t\right)$$
$$\geq \eta_t G_t^{(j)} \cdot \nabla L\left(\bar{\theta}_t\right) - \Delta_{\theta_t - \eta G_t}\left\|\nabla L\left(\bar{\theta}_t\right)\right\|_2$$
$$\geq \eta_t \left\|\nabla L\left(\bar{\theta}_t\right)\right\|_2^2$$
$$- \left(9h_w\sigma'\eta_t + (6l+1)\Delta_{\theta_t}\right)\left\|\nabla L\left(\bar{\theta}_j t\right)\right\|_2.$$

Taking the average over honest parameter servers, dividing by η_t and factoring in the bound on Δ_{θ_t}, we obtain the lemma. □

Theorem 2 (Liveness and safety of parameter servers) *Under our assumptions for any parameter server j, $\langle \mathbb{E}\bar{G}_t, \nabla L\left(\bar{\theta}_t\right)\rangle \geq \frac{3}{4}\left\|\nabla L\left(\theta_t^{(j)}\right)\right\|_2^2$.*

Proof Lemma 17 shows that there exists K such that $\langle \mathbb{E}\bar{G}_t, \nabla L\left(\bar{\theta}_t\right)\rangle \geq \left\|\nabla L\left(\theta_t^{(j)}\right)\right\|_2^2 - K\left\|\nabla L\left(\theta_t^{(j)}\right)\right\|_2$. But by assumption, we know that $K \leq \left\|\nabla L\left(\theta_t^{(j)}\right)\right\|_2/4$. Combining the two inequalities yields the theorem. □

5.4 Bounded moments for *ByzSGD*

Finally, we show three lemmas which are important in the classic convergence proof of SGD [12].

Lemma 18 (Bounded moments of effective gradient) *We show that*

$$\mathbb{E}_{\boldsymbol{\xi}_t|\boldsymbol{\theta}_t} \left\|\bar{G}_t\right\|_2^r \leq A'_r + B'_r\left\|\bar{\theta}_t\right\|_2^r. \tag{36}$$

Proof We define the following notation:

$$\theta_{t+1/2}^{(j)} \triangleq \left(\theta_t^{(j)} - \eta_t G_t^{(j)}\right).$$

We know that $\bar{G}_t = \bar{g}_t - \frac{1}{\eta_t}(\bar{\theta}_{t+1} - \bar{\theta}_{t+1/2})$ and hence we have,

$$\left\|\bar{G}_t\right\|_2 = \left\|\bar{g}_t - \frac{1}{\eta_t}(\bar{\theta}_{t+1} - \bar{\theta}_{t+1/2})\right\|_2 \tag{37}$$
$$\leq \left\|\bar{g}_t\right\|_2 + \frac{1}{\eta_t}\left\|\bar{\theta}_{t+1} - \bar{\theta}_{t+1/2}\right\|_2 \tag{38}$$
$$\leq \left\|\bar{g}_t\right\|_2 + \frac{3}{\eta_t}\Delta_2(\boldsymbol{\theta}_{t+1/2}) \tag{39}$$
$$\leq \left\|\bar{g}_t\right\|_2 + \frac{3}{\eta_t}\left(6d\eta_t h_w\sigma' + (1+3d\eta_t l)\Delta_2(\boldsymbol{\theta}_t)\right), \tag{40}$$

where Δ_2 gives the ℓ_2 diameter. Note that in the last two equations, we use the bound imposed by *MDA* (Lemmas 9), 10, and the triangle inequality of diameters (Lemma 1). Recall Lemma 13, we have

$$\Delta_2\left(\boldsymbol{\theta}_t\right) \leq C\left(1-\frac{\rho}{8}\right)^{t/2} + \frac{24dl\eta_{\lfloor t/2\rfloor}}{\rho} \tag{41}$$

Based on the bounded moments and the bounded variance assumptions, we then have for some constants A and B,

$$\mathbb{E}_{\boldsymbol{\xi}_t|\boldsymbol{\theta}_t} \left\|\bar{G}_t\right\|_2 \leq A + B\left\|\bar{\theta}_t\right\|_2 + C'. \tag{42}$$

We can now define a new constant A′ as follows:

$$A' = A + \frac{3}{\eta_1}\Bigg(6d\eta_1 h_w\sigma' + (1+3d\eta_1 l)$$
$$\left(C\left(1-\frac{\rho}{8}\right)^{t/2} + \frac{24dl\eta_1}{\rho}\right)\Bigg).$$

Thus, we have

$$\underset{\boldsymbol{\xi}_t|\boldsymbol{\theta}_t}{\mathbb{E}} \left\|\bar{G}_t\right\|_2 \leq A' + B \left\|\bar{\theta}_t\right\|_2. \tag{43}$$

Noting that $\|a+b\|_2^r \leq \phi(r)\left(\|a\|_2^r + \|b\|_2^r\right)$, where $\phi(r) = \sum_{i\in\{0\ldots r-1\}} \binom{r}{i}$ then, we have

$$\begin{aligned}\underset{\boldsymbol{\xi}_t|\boldsymbol{\theta}_t}{\mathbb{E}} \left\|\bar{G}_t\right\|_2^r &\leq \left(A' + B\left\|\bar{\theta}_t\right\|_2\right)^r \\ &\leq \phi(r)A'^r + \phi(r)B^r\left\|\bar{\theta}_t\right\|_2^r. \end{aligned}\tag{44}$$

Setting $A'_r = \phi(r)A'^r$ and $B'_r = \phi(r)B^r$ yields the lemma. □

Lemma 19 (Bounded variance to gradient norm) *Noting* $\sigma_t^{(j)} \triangleq \sqrt{\mathbb{E}\left(\left\|g_t^{(j)} - \nabla L\left(\theta_t^{(j)}\right)\right\|_2^2\right)}$ *and* $\sigma_t \triangleq \min_{k\in[h_w]}\left(\sigma_t^{(k)}\right)$*, the norm of gradient at the average of local parameters is lower bounded by:*

$$\frac{\sqrt{8}f_w}{h_w}\sigma_t - Cl\left(1-\frac{\rho}{8}\right)^{t/2} - \frac{24dl^2\eta_{\lfloor t/2\rfloor}}{\rho} \leq \mathbb{E}\left\|\nabla L\left(\bar{\theta}_t\right)\right\|_2. \tag{45}$$

Proof From Lemma 13, we can bound the distance between the average parameter and $\forall j \in [h_{ps}]$ honest parameter as follows:

$$\mathbb{E}\left\|\bar{\theta}_t - \theta_t^{(j)}\right\|_2 \leq C\left(1-\frac{\rho}{8}\right)^{\frac{t}{2}} + \frac{24dl\eta_{\lfloor\frac{t}{2}\rfloor}}{\rho}. \tag{46}$$

By Lipschitz continuity of the loss, $\forall (x,y) \in \left(\mathbb{R}^d\right)^2$:

$$\|\nabla L(x) - \nabla L(y)\|_2 \leq l\,\|x-y\|_2. \tag{47}$$

Then, using the reverse triangular inequality:

$$\begin{aligned}\left|\|\nabla L(x)\|_2 - \|\nabla L(y)\|_2\right| &\leq \|\nabla L(x) - \nabla L(y)\|_2 \\ &\leq l\,\|x-y\|_2 \qquad (48)\\ \Rightarrow \|\nabla L(x)\|_2 &\geq \|\nabla L(y)\|_2 - l\,\|x-y\|_2. \qquad (49)\end{aligned}$$

So, with the honest parameters and linearity of $\mathbb{E}$:

$$\begin{aligned}\mathbb{E}\left\|\nabla L\left(\bar{\theta}_t\right)\right\|_2 &\geq \mathbb{E}\left\|\nabla L\left(\theta_t^{(j)}\right)\right\|_2 - l\mathbb{E}\left\|\bar{\theta}_t - \theta_t^{(j)}\right\|_2 \qquad (50)\\ &\geq \mathbb{E}\left\|\nabla L\left(\theta_t^{(j)}\right)\right\|_2 - Cl\left(1-\frac{\rho}{8}\right)^{\frac{t}{2}} + \frac{24dl^2\eta_{\lfloor\frac{t}{2}\rfloor}}{\rho}. \qquad (51)\end{aligned}$$

By assumption:

$$\begin{aligned}\left\|\nabla L\left(\theta_t^{(j)}\right)\right\|_2 &\geq \frac{\sqrt{8}f_w}{h_w}\sigma_t^{(j)} \geq \frac{\sqrt{8}f_w}{h_w}\sigma_t \qquad (52)\\ &\geq \frac{\sqrt{8}f_w}{h_w}\sigma_t - Cl\left(1-\frac{\rho}{8}\right)^{t/2} - \frac{24dl^2\eta_{\lfloor t/2\rfloor}}{\rho}. \qquad (53)\end{aligned}$$

□

Lemma 20 (Global confinement of the parameter vector) *For any value of* t*, and* $\forall j \in [h_{ps}]$, $\theta_t^{(j)}$ *is almost surely confined within a bounded region.*

Proof Given Lemmas (17, 19), the average effective gradient $\bar{G}$ always points to the same direction of the real gradient, which always points to the origin. Then, given the bound on the parameters diameter (Lemma 13) and the bound on the moments of the effective gradient (Lemma 18), it is evident that correct parameter vectors remain confined beyond a certain horizon D, as a direct consequence of [11]. □

Combining all these lemmas together shows that *ByzSGD* achieves (α, f) Byzantine resilience. We can say more. First, based on Lemma 13, the parameter vectors on correct nodes get closer to each other as t increases, i.e., $\lim_{t\to\infty} \varDelta_2(\boldsymbol{\theta}_t) = 0$. This property enables having the abstraction of a centralized setup (i.e., with one parameter vector), which is well understood and already shown to be converging in the literature. Moreover, based on Lemma 8, we learn that gradients on correct nodes get closer to each other. Second, based on Lemma 14, the effective gradient is always close to the true gradient (while both always pointing to the same direction) and further, the distance between both decreases as t increases. Hence, based on these elements, our algorithm fulfill the requirements of almost-sure convergence as given by [11] (Proposition 2), which is adapted from the classical SGD proof [12] regardless of the identity of the (correct) parameter server.

6 *ByzSGD*: reducing messages with synchrony

We show here that assuming network synchrony, we can boost *ByzSGD*'s performance while keeping the same resilience guarantees. In particular, the number of communicated messages can be reduced as follows: instead of pulling an updated model from q_{ps} servers (line 6 in Algorithm 1), each worker pulls only one model and then checks its legitimacy using two filters: *Lipschitz* and *Outliers* filters. In this case, *ByzSGD* requires $n_w \geq 2f_w + 1$ while keeping $n_{ps} \geq 3f_{ps} + 2$.

6.1 *Lipschitz* filter

Based on the standard Lipschitz continuity of the loss function assumption [11,12], previous work used empirical estimations for the Lipschitz coefficient to filter out gradients from Byzantine workers in asynchronous learning [18]. We use a similar idea, but we now apply it to filter out *models* from Byzantine *servers*. The filter works as follows: consider a worker j that owns a model $\theta_t^{(j)}$ and a gradient it computed $g_t^{(j)}$ based on that model at some step t. A correct server i should include $g_t^{(j)}$ while updating its model $\theta_t^{(i)}$, given network synchrony. Worker j then: (1) estimates the updated model locally $\theta_{t+1}^{(j(l))}$ based on its own gradient and (2) pulls a model $\theta_{t+1}^{(i)}$ from a parameter server i. If server i is correct then the growth of the pulled model $\theta_{t+1}^{(i)}$ (with respect to the local gradient $g_t^{(j)}$) should be close to that of the estimated local model $\theta_{t+1}^{(j(l))}$, based on the guarantees given by *MDA*. Such a growth rate is encapsulated in the *Lipschitz coefficient* of the loss function. If the pulled model is correct then, the worker expects that the Lipschitz coefficient computed based on that model be close to those of the other correct models received before by the worker. Concretely, a worker computes an empirical estimation of the Lipschitz coefficient $k = \frac{\left\| g_{t+1}^{(j)} - g_t^{(j)} \right\|_2}{\left\| \theta_{t+1}^{(j(l))} - \theta_t^{(j)} \right\|_2}$ and then, ensures that it follows the condition $k \leq K_p \triangleq \text{quantile}_{\frac{n_{ps}-f_{ps}}{n_{ps}}} \{K\}$, where K is the list of all previous Lipschitz coefficients k (i.e., with $t_{prev} < t$).

The *Lipschitz* filter, by definition, accepts on average $n_{ps} - f_{ps}$ models for every pulled n_{ps} models. Such a bound makes sense given the round robin fashion of pulling models from servers (by workers) and the existence of (at most) f_{ps} Byzantine servers. Based on this filter, each worker pulls, on average, $n_{ps} + f_{ps}$ each n_{ps} steps. Due to the presence of f_{ps} Byzantine servers, this is a tight lower bound on the communication between each worker and parameter servers to pull the updated model. The worst attack an adversary can do is to send a model that passes the filter (looks like a legitimate model, i.e., very close to a legitimate model) that does not lead to computing a large enough gradient (i.e., leads to minimal learning progress); in other words: an attack that drastically slows down progress. For this reason, such a filter requires $n_{ps} > 3 f_{ps}$. With this bound, the filter ensures the acceptance of at least $f_{ps} + 1$ models for each pulled n_{ps} models, ensuring a majority of correct accepted models anyway and hence, ensuring the progress of learning. Moreover, due to the randomness of choosing the value r (line 3 in Algorithm 3) and the round robin fashion of pulling the models, progress is guaranteed in such a step, as correct and useful models are pulled by other workers, leading to computing correct gradients.

6.2 *Outliers* filter

Although the Lipschitz filter can bound the model growth with respect to gradients, a server can still trick this filter by sending a well-crafted model that is arbitrarily far from the other correct models [7]. To overcome this problem, we use another filter, which we call Outliers *filter*, to bound the distance between models in any two successive steps.

Without loss of generality, consider a correct worker j that pulls models from parameter servers $\theta_t^{(i)} \forall i \in [1 \ldots n_{ps}]$ in a round robin fashion. In each step $t > 1$, the worker computes a local estimation of the next (to be received) model $\theta_t^{(j(l))}$ based on the latest model it has $\theta_{t-1}^{(j)}$ and its own gradient estimation $g_{t-1}^{(j)}$. The local model estimation is done as follows:

$$\theta_t^{(j(l))} = \theta_{t-1}^{(j)} - \eta_{t-1} g_{t-1}^{(j)}. \tag{54}$$

Without loss of generality, assume that worker j pulls the model from some server i in step t. If such a server is correct, it computes the new model $\theta_t^{(i)}$ as follows:

$$\theta_t^{(i)} = \theta_{t-1}^{(i)} - \eta_{t-1} MDA\left(g_{t-1}^{(1)} \ldots g_{t-1}^{(n_w)}\right). \tag{55}$$

Thus, the difference between the local model estimation at worker j and the received model from server i (if it is correct) is given by:

$$\begin{aligned} \left\| \theta_t^{(j(l))} - \theta_t^{(i)} \right\|_2 = & \left\| \left(\theta_{t-1}^{(j)} - \eta_{t-1} g_{t-1}^{(j)}\right) \right. \\ & \left. - \left(\theta_{t-1}^{(i)} - \eta_{t-1} MDA\left(g_{t-1}^{(1)} \ldots g_{t-1}^{(n_w)}\right)\right) \right\| \\ \leq & \left\| \theta_{t-1}^{(j)} - \theta_{t-1}^{(i)} \right\|_2 \\ & + \eta_{t-1} \left\| MDA\left(g_{t-1}^{(1)} \ldots g_{t-1}^{(n_w)}\right) - g_{t-1}^{(j)} \right\|_2. \end{aligned}$$

Based on the guarantees given by *MDA* [24], the following bound holds:

$$\left\| MDA\left(g_t^{(1)} \ldots g_t^{(n_w)}\right) - g_t^{(j)} \right\|_2 \leq \frac{h_w}{\sqrt{8} f_w} \left\| g_t^{(j)} \right\|_2. \tag{56}$$

Based on the results shown before, the maximum distance between two correct models just after a *gather* phase is:

$$\frac{3h_w}{2\sqrt{8} f_w} \eta_{(T \cdot (t \bmod T))} \left\| g_{(T \cdot (t \bmod T))} \right\|_2 T.$$

By induction, we can bound $\left\| \theta_{t-1}^{(j(l))} - \theta_{t-1}^{(i)} \right\|_2$ and hence, we can write:

$$\begin{aligned} \left\| \theta_t^{(j(l))} - \theta_t^{(i)} \right\|_2 \leq & \ \eta_{(T \cdot (t \bmod T))} \left\| g_{(T \cdot (t \bmod T))} \right\|_2 \\ & \times \left(2\Big((t \bmod T) - 1\Big) + \frac{h_w (3T+2)}{2\sqrt{8} f_w} \right). \end{aligned} \tag{57}$$

Thus, a received model $\theta_t^{(i)}$ that satisfies Eq. (57) is considered passing the *Outliers* filter. Such a model is guaranteed to be in the correct cone of models in one *scatter* phase. Note that such a filter cannot be used alone without the *Lipschitz* filter. A Byzantine server can craft a model that satisfies such a filter (i.e., the *Outliers* filter) while being on the opposite direction of minimizing the loss function. Such a model then will be caught by the *Lipschitz* filter.

Combining the *Lipschitz* filter and the *Outliers* filter guarantees that the received model at the worker side is close to the correct one (at the current specific *scatter* phase), representing a reasonable growth, compared to the latest local model at the worker.

Algorithm 3 *ByzSGD*: worker logic (synchronous)

```
1: Calculate the values of $T$ & $seed$
2: model ← init_model(seed)
3: r ← random_int(1,$n_{ps}$)
4: $t \leftarrow 0$
5: grad ← compute_grad(model)
6: repeat
7:    local_model ← apply_grad(model,grad)
8:    if $t$ mod $T = 0$ then
9:       models ← read_models()
10:      model ← Median(models)
11:   else
12:      $i \leftarrow 0$
13:      repeat
14:         new_model ← read_model(($r+t+i$) mod $n_{ps}$)
15:         new_grad ← compute_grad(new_model)
16:         $i \leftarrow i+1$
17:      until pass_filters(new_model)
18:      model ← new_model
19:      grad ← new_grad
20:   end if
21:   $t \leftarrow t+1$
22: until $t >$ max_steps
```

6.3 *ByzSGD*: the synchronous version

Keeping the parameter server algorithm as is (Algorithm 2), Algorithm 3 presents the workers' training loop in the synchronous case. We focus here on the changes in the *ByzSGD* algorithm, compared to the asynchronous case (Sect. 4.2).

In the initialization phase, each worker j chooses a random integer r_j with $1 \leq r_j \leq n_{ps}$ before estimating the gradient at the initial model.

While parameter servers are updating the model (line 7 in Algorithm 2), each worker j speculates the updated model by computing a local view of it, using its local computed gradient and its latest local model (line 7 in Algorithm 3). Then, each worker j pulls *one* parameter vector from server i where, $i = (r_j + t + 1) \bmod n_{ps}$. Such a worker computes the new gradient based on the pulled model. Based on this computation and the local estimate of the updated model, the worker applies the *Lipschitz* and the *Outliers* filters to check the legitimacy of the pulled model. If the model fails to pass the filters, the worker j pulls a new model from the parameter server i_{++}, where $i_{++} = (r_j + t + 2) \bmod n_{ps}$. This process is repeated until a pulled model passes both filters. Every T steps (i.e., in the *gather* phase), each worker j pulls models from *all* servers and aggregates them using *Median*, completing the DMC computation.

Table 2 Models used throughout the evaluation

NN architecture	# parameters	Size (MB)
MNIST_CNN	79510	0.3
CifarNet	1756426	6.7
Inception	5602874	21.4
ResNet-50	23539850	89.8
ResNet-200	62697610	239.2
Transformer	11866786	45.3

7 Experimental evaluation

We implemented our algorithms on top of TensorFlow [1] and PyTorch [51], and we report here on our empirical results, showing the resilience of *ByzSGD* against Byzantine attacks and highlighting the efficiency of its synchronous variant.

7.1 Evaluation setting

7.1.1 Testbed

We use Grid5000 [30] as an experimental platform. We employ up to 20 worker nodes and up to 6 parameter servers. Each node has 2 CPUs (Intel Xeon E5-2630 v4) with 10 cores, 256 GiB RAM, and 2×10 Gbps Ethernet.

7.1.2 Applications

We consider two ML applications for our evaluation: image classification and language modeling. We chose these tasks due to its wide adoption as a benchmark for distributed ML systems, e.g., [16,68]. For the first task, we use MNIST [44] and CIFAR-10 [41] datasets. MNIST consists of handwritten digits. It has 70,000 28×28 images in 10 classes. CIFAR-10 is a widely-used dataset in image classification [58,70]. It consists of 60,000 32×32 colour images in 10 classes. We use image classification as our default application. For the second task, we use the Wikipedia dataset [19] with all the English words as the vocabulary.

For image classification, we employ several NN architectures with different sizes ranging from a small convolutional neural network (CNN) for MNIST, training $<$ 100k param-

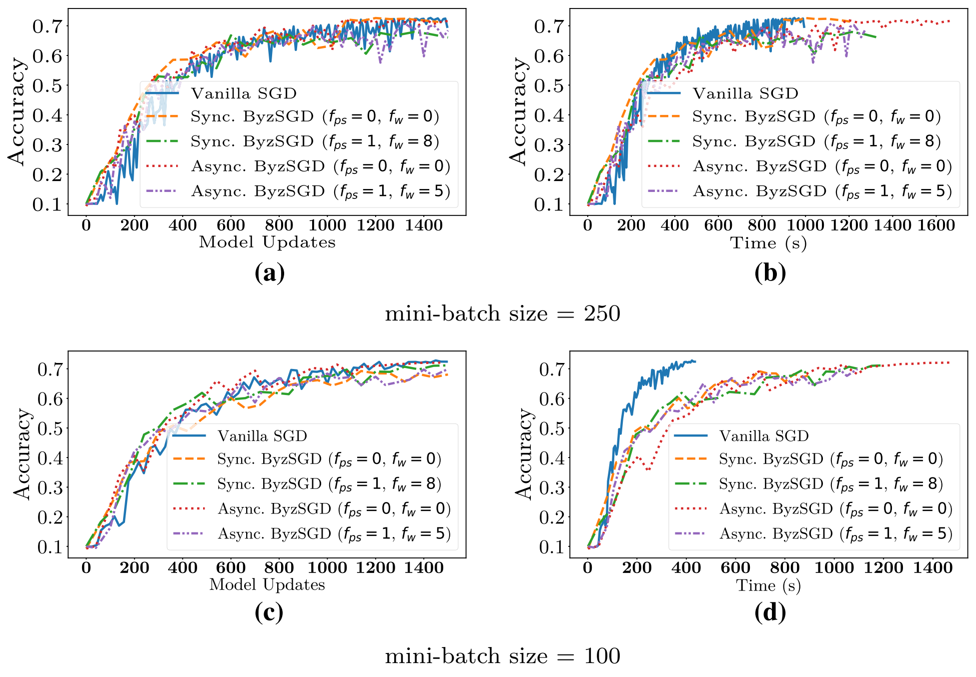


Fig. 3 Convergence in a non-Byzantine environment

eters, to big architectures like ResNet-200 with around 63M parameters (see Table 2). We use *CIFAR10* (as a dataset) and *CifarNet* (as a model) as our default experiment. For the language modeling task, we the Transformer model [61].

7.1.3 Metrics

We evaluate the performance of *ByzSGD* using the following standard metrics.

1. Accuracy (top-1 cross-accuracy) The fraction of correct predictions among all predictions, using the *test* dataset.

2. Loss The total loss value using the *test* dataset. We measure both the accuracy and the loss with respect to time and model updates (or epochs).

3. Throughput The total number of updates that the deployed system can do per second.

7.1.4 Baseline

We consider vanilla SGD as our baseline. Given that such a baseline does not converge in Byzantine environments [17], we use it only to quantify the overhead in non-Byzantine environments.

7.2 Evaluation results

First, we show *ByzSGD*'s performance, highlighting the overhead, in a non-Byzantine environment. Then, we compare the throughput of the synchronous variant to that of the asynchronous variant in a Byzantine-free environment. Finally, we report on the performance of *ByzSGD* in a Byzantine environment, i.e., with Byzantine workers and Byzantine servers. For the Byzantine workers, we show the effect of a recent attack [7] on *ByzSGD*, and then we show the results of 4 different attacks in the case of Byzantine servers. In all experiments, and based on our setup, we use $T = 333$. We discuss the effect of changing T on *ByzSGD*'s performance later in this section.

7.2.1 Non-Byzantine environment

Figure 3 shows the convergence (i.e., progress of accuracy) of all experimented systems with both time and model updates (i.e., training steps). We experiment with two batch sizes and different values for declared Byzantine servers and workers (only for the Byzantine-tolerant deployments). Figure 3a shows that all deployments have almost the same convergence behavior, with a slight loss in final accuracy for the Byzantine-tolerant deployments, which we quantify to around 5%. Such a loss is emphasized with the smaller batch size (Fig. 3c). This accuracy loss is admitted in previous work [65] and is inherited from using statistical methods (basically, *MDA* in our case) for Byzantine resilience. In particular, *MDA* ensures convergence only to a ball around the optimal solution, i.e., local minimum [24]. Moreover, the figures confirm that using a higher batch size gives a more stable convergence for *ByzSGD*. Figure 3a and c show that both variants of *ByzSGD* almost achieve the same convergence.

The cost of Byzantine resilience is more clear when convergence is observed over time (Fig. 3b), especially with the lower batch size (Fig. 3d). We define the convergence

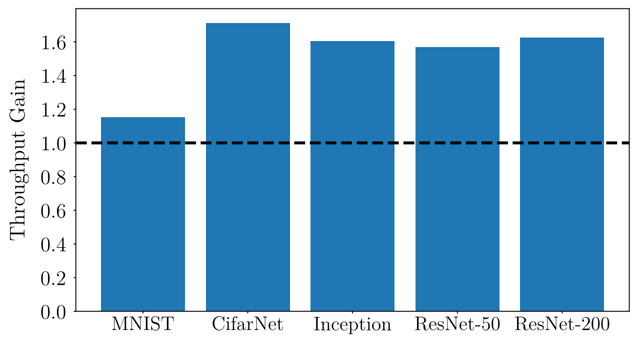


Fig. 4 Throughput gain: sync. relative to async

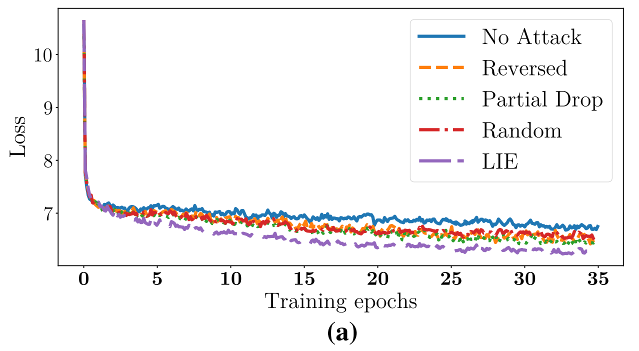
overhead by the ratio of the time taken by *ByzSGD* to reach some accuracy level compared to that taken by vanilla SGD to reach the same accuracy level. For example, in Fig. 3b, vanilla SGD reaches 60% accuracy in 268 s which is around 32% better than the slowest deployment of *ByzSGD*. We draw two main observations from these figures. First, changing the number of declared Byzantine machines affects the progress of accuracy, especially with the asynchronous deployment of *ByzSGD*. This is because servers and workers in such case wait for replies from only $n - f$ machines. Hence, decreasing f forces the receiver to wait for more replies, slowing down convergence. Second, the synchronous variant always outperforms the asynchronous one, especially with non-zero values for declared Byzantine machines, be they servers and workers. Such a result is expected as the synchronous algorithm uses less number of messages per round compared to the asynchronous one. Given that distributed ML systems are network-bound [36,70], reducing the communication overhead significantly boosts the performance (measured by convergence speed in this case) and the scalability of such systems.

7.2.2 Throughput

We do the same experiment again, yet with different state-of-the-art models so as to quantify the throughput gain of the synchronous variant of *ByzSGD*. Figure 4 shows the throughput of synchronous *ByzSGD* normalized by the throughput of the asynchronous *ByzSGD* in each case. From this figure, we see that synchrony helps *ByzSGD* achieve throughput boost (up to 70%) in all cases, where such a boost is emphasized more with large models. This is expected because the main advantage of synchronous *ByzSGD* is to decrease the number of communication messages, where bigger messages are transmitted with bigger models.

7.2.3 Byzantine workers

We study here the convergence of *ByzSGD* in the presence of Byzantine workers. We experiment first with different variants of workers' misbehavior using our language modelling application. Such variants include message drops, random gradients, or reversed gradients; such misbehavior are typically used to test Byzantine-resilient GARs, e.g., [65]. Then, we focus on a more recent attack, using the image classification task, that is coined as *A little is enough attack* (LIE) [7]. This attack focuses on changing each dimension in gradients of Byzantine workers to trick some of Byzantine-resilient GARs, e.g., [11,24].



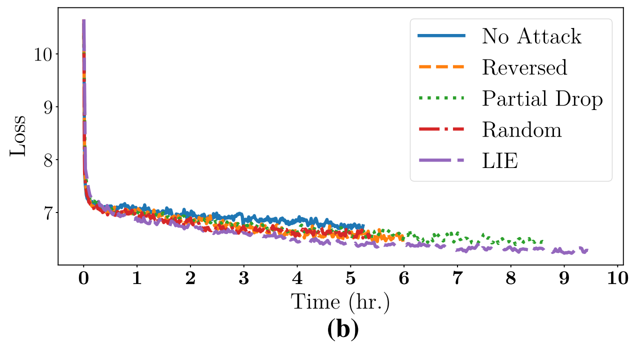
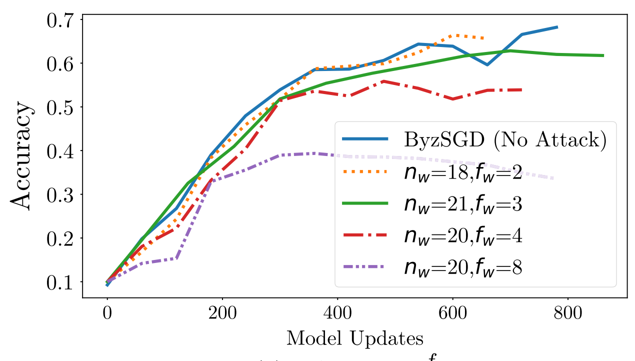


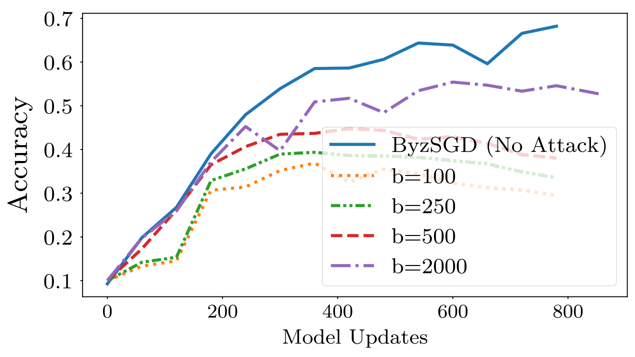
Fig. 5 Convergence of training, using our language modelling task, in the presence of Byzantine workers

Figure 5 shows the progress of the loss with different kinds of attacks compared to the vanilla case with no attacks. In all cases, we employ 3 servers with $f_{ps} = 1$ (but with no actual attack) and 10 workers with $f_w = 1$ (with the attack starting from the beginning of training). Figure 5a shows that *ByzSGD* can tolerate all the experimented Byzantine attacks: all deployments achieve comparable loss values after 35 epochs. This indicates the effectiveness of our GARs against such attacks and confirms empirically the Byzantine resilience of *ByzSGD*. Figure 5b shows that although Byzantine behavior can be tolerated, this resilience comes at the cost of delayed convergence. We can see that simple behavior like putting random data or reversing the gradient would have lower effect on delaying convergence compared to a stronger attack like LIE.

We focus now on the *LIE* attack. We apply this attack to multiple deployments of *ByzSGD*. In each scenario, we apply the strongest possible change in gradients' coordinates so as



(a) The ratio $\frac{f_w}{n_w}$



(b) Batch size

Fig. 6 Convergence in the presence of Byzantine workers

to hamper the convergence the most. We study the effect of this attack on the convergence of *ByzSGD* with both the ratio of Byzantine workers to the total number of workers (Fig. 6a) and the batch size (Fig. 6b). We use the deployment with no Byzantine behavior (*No Attack*) as a baseline.

Figure 6a shows that the effect of the attack starts to appear clearly when the number of Byzantine workers is a significant fraction (more than 20%) of the total number of workers. This is intuitive as the attack tries to increase the variance between the submitted gradients to the parameter servers and hence, increases the ball (around the local minimum) to which the used GAR converges (see e.g., [11,24] for a theoretical analysis of the interplay between the variance and the Byzantine resilience). Stretching the number of Byzantine workers to the maximum ($f_w = 8$) downgrades the accuracy to around 40% (compared to 67% in "No Attack" case). This can be explained by the large variance between honest gradients, above what *MDA* requires, as we discuss in Sect. 7.

Increasing the batch size not only improves the accuracy but also the robustness of *ByzSGD* (by narrowing down the radius of the ball around the convergence point, where the model will fluctuate as proven in [11,24]). Figure 6b fixes the ratio of f_w to n_w to the biggest allowed value to see the effect of using a bigger batch size on the convergence behavior. This figure confirms that increasing the batch size increases the robustness of *ByzSGD*. Moreover, based on our experiments, setting 25% of workers to be Byzantine while using a batch size of (up to) 256 does not experimentally satisfy the assumption on the variance of *MDA* in this deployment, which leads to a lower accuracy after convergence.

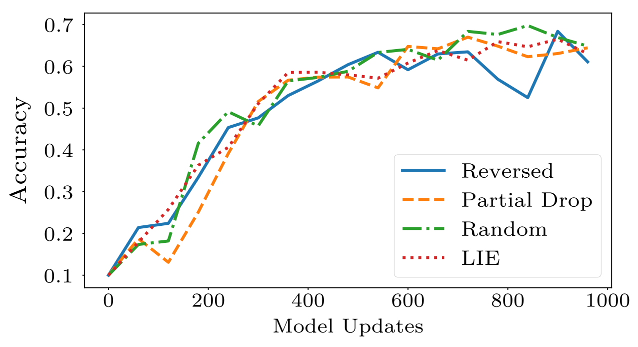


Fig. 7 Convergence with one Byzantine server

7.2.4 Byzantine servers

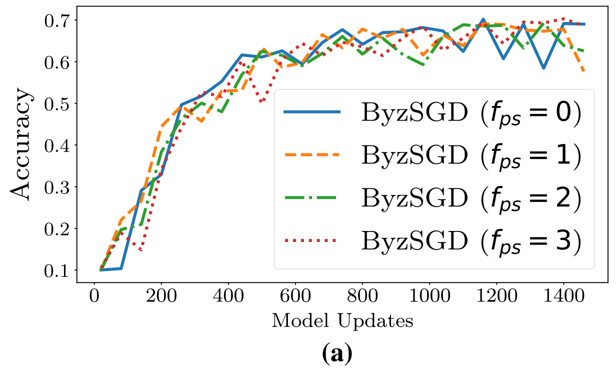
Figure 7 shows the convergence of *ByzSGD* in the presence of 1 Byzantine server. We experimented with 4 different adversarial behaviors: (1) *Reversed:* the server sends a correct model multiplied by a negative number, (2) *Partial Drop:* the server randomly chooses 10% of the weights and set them to zero (this simulates using unreliable transport protocol in the communication layer, which was proven beneficial in some cases, e.g., [17]), (3) *Random:* the server replaces the learned weights by random numbers, and (4) *LIE*, an attack inspired from the *little is enough* attack [7], in which the server multiplies each of the individual weights by a small number z, where $|z - 1| < \delta$ with δ close to zero; $z = 1.035$ in our experiments. Such a figure shows that *ByzSGD* can tolerate the experimented Byzantine behavior and guarantee the learning convergence to a high accuracy.

7.2.5 Convergence with multiple Byzantine servers

Figure 8 shows the convergence with different numbers of *declared* Byzantine servers. To allow for (up to) 3 Byzantine servers, we use a total of 10 servers in this experiment. Such figures show that changing the number of Byzantine servers does not affect the the number of steps required for convergence (Fig. 8a). Yet, deployments with a higher value for f_{ps} require slightly more time to converge. Note that in this experiment we have not employed real attacks nor changed the total number of machines (for both servers and workers) used.

7.2.6 Effect of changing T

The value of T denotes the number of steps done in one *scatter* phase (i.e., before entering one *gather* phase). Figure 9



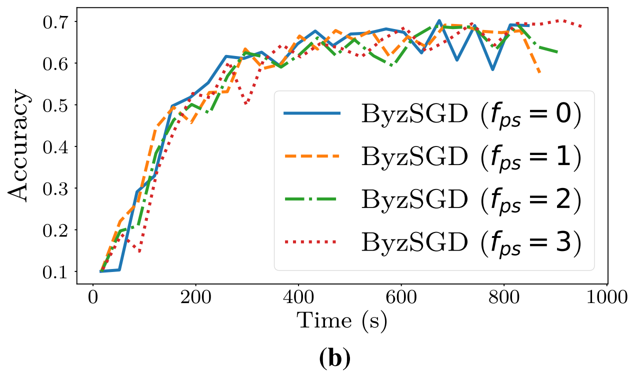


Fig. 8 The effect of changing f_{ps} on performance

shows the effect of changing the value of T on convergence with both time and model updates in both a Byzantine and a Byzantine-free environments. Figure 9a shows that the value of T almost does not have any effect on the convergence w.r.t. the model updates. This happens because models on correct servers almost do not drift from each other (as all the servers are correct). Interestingly, Fig. 9b shows that using a higher value for T helps converge faster. This is because increasing T decreases the communication overhead, achieving faster updates and higher throughput. However, it is important to note that as the value of T increases, the expected drifts between models on correct servers increases, and it becomes easier for the Byzantine server to trick the workers. Figure 9c and d show the convergence with different values for T under the *Reversed* attack (in which the Byzantine machine reverses the direction of the correct vector). Though setting $T = 1$ slows down the convergence, this case shows the most stable convergence behavior (especially towards the end when approaching a local minimum; see Fig. 9c). Yet, higher values for T lead to increased noise and oscillations towards the end of convergence. Notably, testing with $T \leq 333$ is safe in this setup and that is why convergence is reached in all cases.

7.2.7 Filters false negatives

ByzSGD's filters (in the synchronous case) may have false negatives, i.e., falsely-reject correct models (at workers) from correct servers. This leads to wasted communication bandwidth and possibly slows down the convergence. We observe the number of rejected models on workers after 500 learning steps. Figure 10 shows the ratio of false negatives to the total number of submitted models in different scenarios.

Figure 10a shows such a ratio with different values for T while no attack is employed yet, with $f_{ps} = 1$. In general, the ratio of false negatives never exceeds 1% in this experiment, and it is almost stable with increasing T. Note that 333 is the maximum value allowed for T in this setup (to follow the safety rules of *ByzSGD*). With $T = 1$, the false negatives are

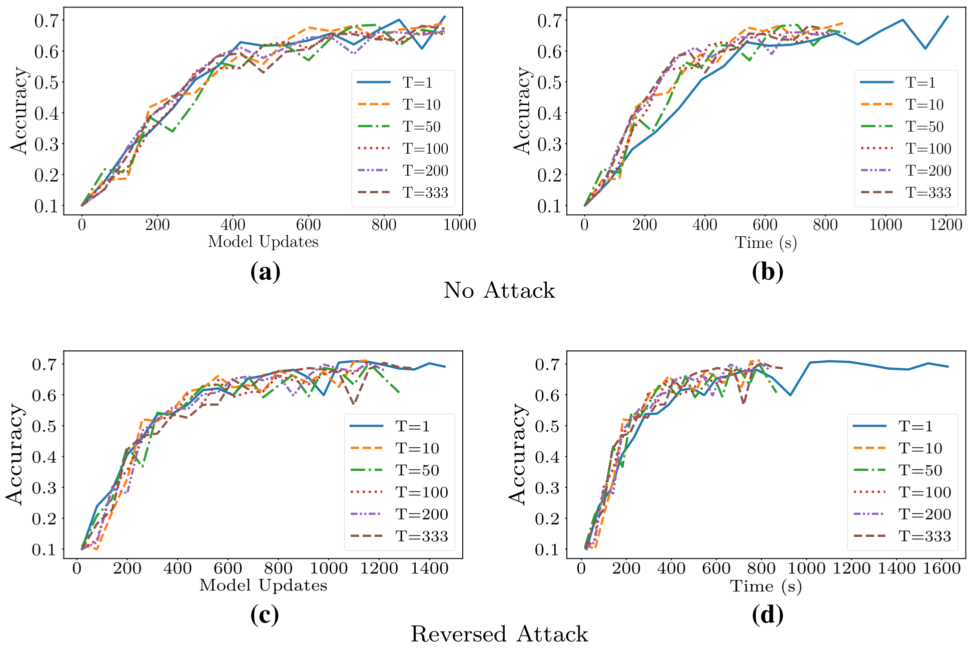


Fig. 9 The effect of changing T on performance

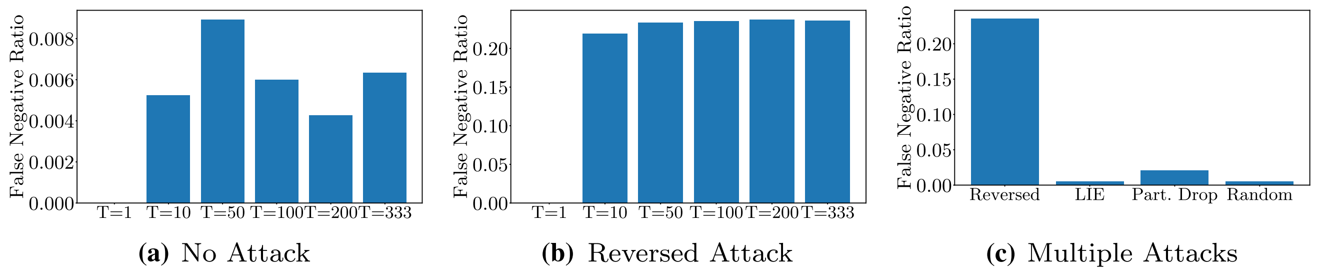


Fig. 10 False negative percentage with different scenarios

always zero, simply because the filters do not work in this setup (i.e., the *gather* phase is entered in every step). Such a figure shows that our filtering mechanism is effective in not producing many false negatives and hence, do not waste communication rounds (when a model is rejected, the worker asks for another model from a different server).

We repeated the same experiment yet with employing the *Reversed* attack from the Byzantine server; results in Fig. 10b. Such an attack is effective (in terms of bandwidth waste), especially with $T \geq 50$. Yet, the wasted bandwidth is upper bounded by 25% in all cases, which is the ratio of the number of Byzantine servers to the total number of servers 1:4. Figure 10c shows that other attacks are not that effective in wasting the bandwidth, as the filters can successfully filter out only the Byzantine models and accept the correct ones. Other than the *Reversed* attack, the false negatives ratio in this experiment do not exceed 3.5%.

7.2.8 Validating the bounded gradients variance to norm ratio assumption

To make progress at every step, any Byzantine-resilient GAR, that is based on statistical robustness, requires a bound on the ratio $\frac{variance}{norm}$ of the correct gradient estimations. Intuitively, not having such a bound would allow the correct gradients to become *indistinguishable* from some random noise. This is problematic, since such GARs, e.g., [11,24,59,66] rely on techniques analogous to *voting* (i.e., median-like techniques in high-dimension): if the correct majority does not agree (appears "random"), then the Byzantine minority controls the aggregated gradient. For example, not satisfying these bounds makes the used GARs vulnerable against recently-proposed attacks like *A little is enough* attack [7] (see Fig. 6). Such a bound is to ensure that, no matter the received Byzantine gradients, the expected value of the aggregated gradient does lie in the same half-space as the real gradient, leading for every step taken to more *optimal* parameters (smaller loss).

Here, we try to understand when this assumed bound on the variance to norm ratio (Eq. (3)) holds and when it does not. The most straightforward way to fulfill such an assumption is to increase the batch size used for training. The question is then what the minimum batch size (that can be used while satisfying such a bound) is, and whether it is small enough for the distribution of the training to still make sense.

Methodology. We use the same setup and hyperparameters as used in our evaluation of *ByzSGD*. We estimate over the first 100 steps of training the variance to norm ratio of correct gradient estimations for several batch sizes. We plot the average (line) and standard deviation (error bar) of these ratios over these 100 steps (Fig. 11). We show the bound required by two Byzantine-resilient GAR: *MDA*, and *Multi-Krum* [11]. We find *Multi-Krum* a very good example on a widely-used GAR that unfortunately does not seem to provide any practical[9] guarantee, due to its unsatisfied assumption.[10] We also experimented with two values of the number of declared Byzantine workers: $f = 1, 5$. Increasing the value of f calls for a tighter bound on the variance to norm ratio.

Results. Figure 11 depicts the relation between the variance (of gradients) to norm ratio with the batch size. According to such a figure, *Multi-Krum* cannot be safely used even with the largest experimented batch size, i.e., 256. Otherwise, the variance bound assumption such a GAR builds on is not satisfied and hence, an adversary can break its resilience guarantees [7]. *MDA* gives a better bound on the variance, which makes it more practical in this sense: typical batch size of 128 for example can be safely used with $f = 1$. However, *MDA* is not safe to use with $f = 5$ even with the largest experimented batch size ($b = 256$). This is confirmed in our experiments, where we show that an adversary can use such a vulnerability (due to the unsatisfied assumption) to reduce the learning accuracy. Having the optimal bound on variance while guaranteeing Byzantine resilience and convergence remains an open question.

[9] At least on our academic model and dataset.

[10] It needs very low variance to norm ratio of correct gradient estimations, e.g. 0.08 for $(n, f) = (18, 5)$.

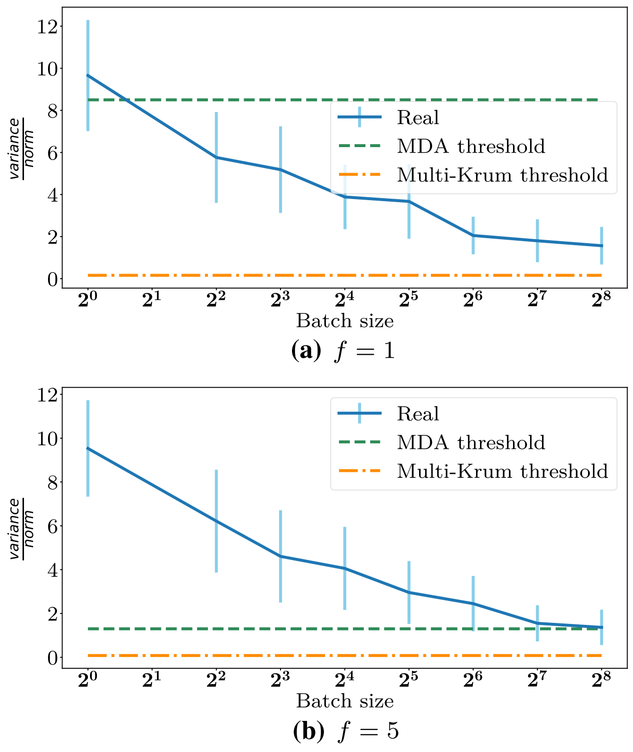


Fig. 11 Variance to norm ratio with different batch sizes, compared to the bound/threshold required by two Byzantine-resilient GARs: *MDA* and *Multi-Krum*. To satisfy a GAR assumption/condition, the *real* $\frac{variance}{norm}$ value should be lower than the GAR bound. For instance, *MDA* can be used with batch-size=32 with $f = 1$, but not with $f = 5$ (as the *real* $\frac{variance}{norm}$ value is higher than the *MDA* threshold)

8 Concluding remarks

8.1 Summary

This paper is the first step towards genuinely distributed Byzantine-resilient Machine Learning (ML) solutions that do not trust any network component nor assume any bound on communication or computation delays. We present *ByzSGD* that guarantees Byzantine resilience, to both Byzantine servers and workers, in the case of i.i.d. local data distributions. Through the introduction of a series of novel ideas, the *Scatter/Gather* protocol, the *Distributed Median-based Contraction (DMC)* module, and the filtering mechanisms, we show that *ByzSGD* works in an asynchronous setting, and we show how we can leverage synchrony to boost performance. We built *ByzSGD* on top of both TensorFlow and PyTorch, and we show that it tolerates Byzantine behavior with 32% overhead compared to vanilla SGD.

8.2 Related work

With the impracticality (and sometimes impossibility [28]) of applying exact consensus to ML applications, the approximate consensus [27] seems to be a good candidate. In approximate consensus, all nodes try to decide values that are arbitrarily close to each other and that are within the range of values proposed by correct nodes. Several approximate consensus algorithms were proposed with different convergence rates, communication/computation costs, and supported number of tolerable Byzantine nodes, e.g., [2,21, 26,48].

Inspired by approximate consensus, several Byzantine-resilient ML algorithms were proposed yet, all assumed a single correct parameter server: only workers could be Byzantine. Three Median-based aggregation rules were proposed to resist Byzantine attacks [65]. Krum [11] and Multi-Krum [17] used a distance-based algorithm to eliminate Byzantine inputs and average the correct ones. Bulyan [24] proposed a meta-algorithm to guarantee Byzantine resilience against a strong adversary that can fool the aforementioned aggregation rules. Draco [15] used coding schemes and redundant gradient computation for Byzantine resilience, where Detox [52] combined coding schemes with Byzantine-resilient aggregation for better resilience and overhead guarantees. Kardam [18] uses filtering mechanisms to tolerate Byzantine workers in an asynchronous learning setup. *ByzSGD* augments these efforts by tolerating Byzantine servers in addition to Byzantine workers.

8.3 Open questions

This paper opens several interesting questions.

First, the relation between the frequency of entering the *gather* phase (i.e., the value of T) and the variance between models on correct servers is both data and model dependent. In our analysis, we provide safety guarantees on this relation that always ensure Byzantine resilience. However, we believe that in some cases, entering the *gather* phase more frequently may lead to a noticeable improvement in the convergence speed. The trade-off between this gain and the corresponding communication overhead is an interesting open question.

Second, the Lipschitz filter requires $n_{ps} > 3f_{ps}$. There is another tradeoff here between the communication overhead and the required number of parameter servers. One can use Byzantine-resilient aggregation of models, which requires only $n_{ps} > 2f_{ps}$, yet requires communicating with *all* servers in each step. In our design, we strive for reducing the communication overhead, given that communication is the bottleneck [36,70].

Third, given that the distributed ML problem is communication bound and that *ByzSGD* introduces additional communication overhead, it is now interesting to explore the interplay between compression and/or quantization techniques with Byzantine resilience.

Fourth, we leave open the question of designing a Byzantine-resilient algorithm that have desirable privacy preservation properties. Such an algorithm would be useful in cases where privacy of workers should not be compromised by the servers in an environment with Byzantine servers and workers.

Fifth, *ByzSGD* assumes synchronous SGD. Following the asynchronous SGD regime opens new challenges and attacks. For example, in such a regime, a strong adversary can control the order of messages the different nodes receive, possibly hindering models on such nodes to contract. The question of how to ensure contractness and convergence in such a case is also left open.

Acknowledgements This work has been supported in part by the Swiss National Science Foundation (FNS grant 200021_182542/1). Most experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

Funding Open access funding provided by EPFL Lausanne.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., et al. Tensorflow: a system for large-scale machine learning. In: 12th *USENIX* Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 265–283 (2016)
2. Abraham, I., Amit, Y., Dolev, D.: Optimal resilience asynchronous approximate agreement. In: International Conference on Principles of Distributed Systems, pp. 229–239. Springer (2004)
3. Alain, G., Lamb, A., Sankar, C., Courville, A., Bengio, Y.: Variance reduction in SGD by distributed importance sampling (2015). arXiv preprint arXiv:1511.06481
4. Alistarh, D., Allen-Zhu, Z., Li, J.: Byzantine stochastic gradient descent. In: Advances in Neural Information Processing Systems, pp. 4613–4623 (2018)
5. Alistarh, D., Li, J., Tomioka, R., Vojnovic, M.: QSGD: randomized quantization for communication-optimal stochastic gradient descent (2016). arXiv preprint arXiv:1610.02132
6. Bartlett, P.L., Foster, D.J., Telgarsky, M.J.: Spectrally-normalized margin bounds for neural networks. In: Neural Information Processing Systems, pp. 6241–6250 (2017)
7. Baruch, G., Baruch, M., Goldberg, Y.: A little is enough: circumventing defenses for distributed learning. In: Advances in Neural Information Processing Systems, 32 (2019)
8. Bernstein, J., Zhao, J., Azizzadenesheli, K., Anandkumar, A.: SignSGD with majority vote is communication efficient and fault tolerant (2018). arXiv preprint arXiv:1810.05291
9. Biggio, B., Nelson, B., Laskov, P.: Poisoning attacks against support vector machines (2012). arXiv preprint arXiv:1206.6389
10. Biggio, B., Roli, F.: Wild patterns: ten years after the rise of adversarial machine learning. Pattern Recognit. **84**, 317–331 (2018)
11. Blanchard, P., El Mhamdi, E.M., Guerraoui, R., Stainer, J.: Machine learning with adversaries: byzantine tolerant gradient descent. In: Neural Information Processing Systems, pp. 118–128 (2017)
12. Bottou, L.: Online learning and stochastic approximations. Online Learn. Neural Netw. **17**(9), 142 (1998)
13. Cachin, C., Guerraoui, R., Rodrigues, L.: Introduction to Reliable and Secure Distributed Programming. Springer, Berlin (2011)
14. Castro, M., Liskov, B., et al.: Practical Byzantine fault tolerance. In: USENIX Symposium on Operating Systems Design and Implementation, vol. 99, pp. 173–186 (1999)
15. Chen, L., Wang, H., Charles, Z., Papailiopoulos, D.: Draco: byzantine-resilient distributed training via redundant gradients. In: International Conference on Machine Learning, pp. 902–911 (2018)
16. Chilimbi, T.M., Suzue, Y., Apacible, J., Kalyanaraman, K.: Project adam: building an efficient and scalable deep learning training system. In: USENIX Symposium on Operating Systems Design and Implementation, vol. 14, pp. 571–582 (2014)
17. Damaskinos, G., El Mhamdi, E.M., Guerraoui, R., Guirguis, A., Rouault, S.: Aggregathor: byzantine machine learning via robust gradient aggregation. In: The Conference on Systems and Machine Learning (MLSys) (2019)
18. Damaskinos, G., El Mhamdi, E.M., Guerraoui, R., Patra, R., Taziki, M.: Asynchronous byzantine machine learning (the case of SGD). In: International Conference on Machine Learning, pp. 1153–1162 (2018)
19. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2018). arXiv preprint arXiv:1810.04805
20. Diakonikolas, I., Kamath, G., Kane, D.M., Li, J., Moitra, A., Stewart, A.: Robustly learning a gaussian: getting optimal error, efficiently. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 2683–2702 (2018)
21. Dolev, D., Lynch, N.A., Pinter, S.S., Stark, E.W., Weihl, W.E.: Reaching approximate agreement in the presence of faults. J. ACM JACM **33**(3), 499–516 (1986)
22. El-Mhamdi, E.-M., Farhadkhani, S., Guerraoui, R., Guirguis, A., Hoang, L.N., Rouault, S.: Collaborative learning in the jungle. In: Advances in Neural Information Processing Systems (2021)
23. El-Mhamdi, E.-M., Guerraoui, R., Guirguis, A., Hoang, L.N., Rouault, S.: Genuinely distributed Byzantine machine learning. In: Proceedings of the 39th Symposium on Principles of Distributed Computing, pp. 355–364 (2020)
24. El-Mhamdi, E.-M., Guerraoui, R., Rouault, S.: The hidden vulnerability of distributed learning in byzantium. In: International Conference on Machine Learning, pp. 3521–3530 (2018)
25. Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., Thrun, S.: Dermatologist-level classification of skin cancer with deep neural networks. Nature **542**(7639), 115 (2017)
26. Fekete, A.D.: Asymptotically optimal algorithms for approximate agreement. In: Proceedings of the Fifth Annual ACM Symposium on Principles of Distributed Computing, pp. 73–87 (1986)
27. Fekete, A.D.: Asynchronous approximate agreement. In: Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing, pp. 64–76 (1987)
28. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. J. ACM (JACM) **32**(2), 374–382 (1985)
29. Gilmer, J., Metz, L., Faghri, F., et al. Adversarial spheres (2018). arXiv preprint arXiv:1801.02774
30. Grid5000. Grid5000. https://www.grid5000.fr/
31. Guerraoui, R., Guirguis, A., Plassmann, J., Ragot, A., Rouault, S.: Garfield: system support for Byzantine machine learning. In: 2021

51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 39–51. IEEE (2021)
32. Guo, S., Zhang, T., Xie, X., Ma, L., Xiang, T., Liu, Y.: Towards byzantine-resilient learning in decentralized systems (2020). arXiv preprint arXiv:2002.08569
33. Hashemi, H., Wang, Y., Guo, C., Annavaram, M.: Byzantine-robust and privacy-preserving framework for FEDML (2021). arXiv preprint arXiv:2105.02295
34. He, L., Karimireddy, S.P., Jaggi, M.: Secure byzantine-robust machine learning (2020). arXiv preprint arXiv:2006.04747
35. Hecht-Nielsen, R.: Theory of the backpropagation neural network. In: Neural Networks for Perception, pp. 65–93. Elsevier (1992)
36. Hsieh, K., Harlap, A., Vijaykumar, N., et al.: Gaia: Geo-distributed machine learning approaching LAN speeds. In: USENIX Symposium on Networked Systems Design and Implementation, pp. 629–647 (2017)
37. Karimireddy, S.P., He, L., Jaggi, M.: Byzantine-robust learning on heterogeneous datasets via resampling (2020). arXiv preprint arXiv:2006.09365
38. Karimireddy, S.P., He, L., Jaggi, M.: Learning from history for byzantine robust optimization. In: Meila, M., Zhang, T., (eds) Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18–24 July 2021, Virtual Event, volume 139 of Proceedings of Machine Learning Research, pp. 5311–5319. PMLR (2021)
39. Kim, L. How many ads does google serve in a day? (2012). http://goo.gl/oIidXO
40. Konečnỳ, J., McMahan, B., Ramage, D.: Federated optimization: distributed optimization beyond the datacenter (2015). arXiv preprint arXiv:1511.03575
41. Krizhevsky, A., Nair, V., Hinton, G.: Cifar dataset. https://www.cs.toronto.edu/~kriz/cifar.html
42. Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Trans. Program. Lang. Syst. TOPLAS **4**(3), 382–401 (1982)
43. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
44. Lecunn, Y.: MNIST dataset (1998). http://yann.lecun.com/exdb/mnist/
45. Li, M., Andersen, D.G., Park, J.W., et al.: Scaling distributed machine learning with the parameter server. In: 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), pp. 583–598 (2014)
46. Li, M., Zhou, L., Yang, Z., et al.: Parameter server for distributed machine learning. In: Big Learning NeurIPS Workshop, vol. 6, pp. 2 (2013)
47. Liu, S., Gupta, N., Vaidya, N.H.: Approximate byzantine fault-tolerance in distributed optimization. In: Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing, pp. 379–389 (2021)
48. Mendes, H., Herlihy, M., Vaidya, N.H., Garg, V.K.: Multidimensional agreement in Byzantine systems. Distrib. Comput. **28**(6), 423–441 (2015)
49. Meng, X., Bradley, J., Yavuz, B., et al.: Mllib: machine learning in apache spark. J. Mach. Learn. Res. **17**(1), 1235–1241 (2016)
50. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Asia Conference on Computer and Communications Security, pp. 506–519 (2017)
51. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: an imperative style, high-performance deep learning library. Adv. Neural Inf. Process. Syst. **32**, 8026–8037 (2019)
52. Rajput, S., Wang, H., Charles, Z., Papailiopoulos, D.: Detox: a redundancy-based framework for faster and more robust gradient aggregation. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
53. Rosasco, L., De Vito, E., Caponnetto, A., Piana, M., Verri, A.: Are loss functions all the same? Neural Comput. **16**(5), 1063–1076 (2004)
54. Rousseeuw, P.J.: Multivariate estimation with high breakdown point. Math. Stat. Appl. **8**, 283–297 (1985)
55. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
56. Schneider, F.B.: Implementing fault-tolerant services using the state machine approach: a tutorial. ACM Comput. Surv. **22**(4), 299–319 (1990)
57. So, J., Güler, B., Avestimehr, A.S.: Byzantine-resilient secure federated learning. IEEE J. Sel. Areas Commun. **39**, 2168–2181 (2020)
58. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
59. Su, L.: Defending distributed systems against adversarial attacks: consensus, consensus-based learning, and statistical learning. PhD thesis, University of Illinois at Urbana-Champaign (2017)
60. Su, L., Shahrampour, S.: Finite-time guarantees for byzantine-resilient distributed state estimation with noisy measurements. IEEE Trans. Autom. Control **65**(9), 3758–3771 (2019)
61. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
62. Virmaux, A., Scaman, K.: Lipschitz regularity of deep neural networks: analysis and efficient estimation. In: Advances in Neural Information Processing Systems, pp. 3835–3844 (2018)
63. Vyavahare, P., Su, L., Vaidya, N.H.: Distributed learning with adversarial agents under relaxed network condition (2019). arXiv preprint arXiv:1901.01943
64. Xiao, H., Biggio, B., Brown, G., Fumera, G., Eckert, C., Roli, F.: Is feature selection secure against training data poisoning? In: International Conference on Machine Learning, pp. 1689–1698 (2015)
65. Xie, C., Koyejo, and O., Gupta, I.: Generalized Byzantine-tolerant SGD (2018). arXiv preprint arXiv:1802.10116
66. Xie, C., Koyejo, O., Gupta, I.: Phocas: dimensional byzantine-resilient stochastic gradient descent (2018). arXiv preprint arXiv:1805.09682
67. Xie, C., Koyejo, O., Gupta, I.: Zeno: Byzantine-suspicious stochastic gradient descent (2018). arXiv preprint arXiv:1805.10032
68. Xu, H., Ho, C-Y., Abdelmoniem, A.M., Dutta, A., Bergou, E.H., Karatsenidis, K., Canini, M., Kalnis, P.: Grace: a compressed communication framework for distributed machine learning. In 2021 IEEE 41st international conference on distributed computing systems (ICDCS), pp. 561–572. IEEE (2021)
69. Yang, H., Zhang, X., Fang, M., Liu, J.: Byzantine-resilient stochastic gradient descent for distributed learning: a lipschitz-inspired coordinate-wise median approach. In: 2019 IEEE 58th Conference on Decision and Control (CDC), pp. 5832–5837. IEEE (2019)
70. Zhang, H., Zheng, Z., Xu, S., Dai, W., Ho, O., Liang, X., Hu, Z., Wei, J., Xie, P., Xing, E.P.: Poseidon: an efficient communication architecture for distributed deep learning on GPU clusters. In: USENIX Annual Technical Conference, pp. 181–193 (2017)
71. Zhang, S., Choromanska, A.E., LeCun, Y.: Deep learning with elastic averaging SGD. In: Neural Information Processing Systems, pp. 685–693 (2015)