# BYZANTINE-RESILIENT NON-CONVEX STOCHASTIC GRADIENT DESCENT


## ABSTRACT

We study adversary-resilient stochastic distributed optimization, in which $m$ machines can independently compute stochastic gradients, and cooperate to jointly optimize over their local objective functions. However, an $\alpha$-fraction of the machines are *Byzantine*, in that they may behave in arbitrary, adversarial ways. We consider a variant of this procedure in the challenging *non-convex* case. Our main result is a new algorithm `SafeguardSGD` which can provably escape saddle points and find approximate local minima of the non-convex objective. The algorithm is based on a new concentration filtering technique, and its sample and time complexity bounds match the best known theoretical bounds in the stochastic, distributed setting when no Byzantine machines are present. Our algorithm is practical: it improves upon the performance of prior methods when training deep neural networks, it is relatively lightweight, and is the first method to withstand two recently-proposed Byzantine attacks.


## 1 INTRODUCTION

Motivated by the pervasiveness of large-scale distributed machine learning, there has recently been significant interest in providing distributed optimization algorithms with strong *fault-tolerance* guarantees. In this context, the strongest, most stringent fault model is that of *Byzantine faults* (Lamport et al., 1982): given $m$ machines, each having access to private data, at most an $\alpha$ fraction of the machines can behave in arbitrary, possibly adversarial ways, with the goal of breaking or at least slowing down the algorithm. Although extremely harsh, this fault model is the "gold standard" in distributed computing (Lynch, 1996; Lamport et al., 1982; Castro et al., 1999), as algorithms proven to be correct in this setting are guaranteed to converge under arbitrary system behaviour.

A setting of particular interest in this context has been that of distributed *stochastic* optimization. Here, the task is to minimize some stochastic function $f(x) = \mathbb{E}_{s\sim\mathcal{D}}[f_s(x)]$ over a distribution $\mathcal{D}$, where $f_s(\cdot)$ can be viewed as the loss function for sample $s \sim \mathcal{D}$. We assume there are $m$ machines (workers) and an honest master, and $\alpha < 1/2$ fraction of the workers may be Byzantine. In each iteration $t$, each worker has access to a version of the global iterate $x_t$, which is maintained by the master. The worker can independently sample $s \sim \mathcal{D}$, compute $\nabla f_s(x_t)$, and then synchronously send this stochastic gradient to the master. The master aggregates the workers' messages, and sends an updated iterate $x_{t+1}$ to all the workers. Eventually, the master has to output an approximate minimizer of $f$. Clearly, the above description only applies to *honest* workers; Byzantine workers may deviate arbitrarily and return adversarial "gradient" vectors to the master in every iteration.

This distributed framework is quite general and well studied. One of the first references in this setting studied distributed PCA and regression (Feng et al., 2014). Other early approaches (Blanchard et al., 2017; Chen et al., 2017; Su and Vaidya, 2016a;b; Xie et al., 2018a) relied on defining generalizations of the geometric median. These approaches can withstand up to half of the nodes being malicious, but can have relatively high local computational cost $\Omega(m^2 d)$ (Blanchard et al., 2017; Chen et al., 2017), where $m$ is the number of nodes and $d$ is the problem dimension, and usually have sub-optimal sample and iteration complexities.

Follow-up work resolved this last issue when the objective $f(\cdot)$ is *convex*, leading to tight sample complexity bounds. Specifically, Yin et al. (2018) provided bounds for *gradient descent*-type algorithms, and showed that the bounds are tight when the dimension is constant. Alistarh et al. (2018) provided a stochastic gradient descent (SGD) type algorithm and showed that its sample and time complexities are asymptotically optimal even when the dimension is large.

**Non-convex Byzantine-resilient stochastic optimization.** In this paper, we focus on the more challenging non-convex setting, and shoot for the strong goal of finding *approximate local minima* (a.k.a. second-order critical points). In a nutshell, our main result is the following. Fix $d$ to denote the dimension, and let the objective $f : \mathbb{R}^d \to \mathbb{R}$ be Lipschitz smooth and second-order smooth.

We have $m$ worker machines, each having access to unbiased, bounded estimators of the gradient of $f$. Given an initial point $x_0$, the `SafeguardSGD` algorithm ensures that, even if at most $\alpha < 1/2$ fraction of the machines are Byzantine, after

$$T = \widetilde{O}\left(\left(\alpha^2 + \frac{1}{m}\right)\frac{d(f(x_0) - \min f(x))}{\varepsilon^4}\right) \qquad \text{parallel iterations,}$$

for at least a constant fraction of the indices $t \in [T]$, the following hold:

$$\|\nabla f(x_t)\| \leq \varepsilon \quad \text{and} \quad \nabla^2 f(x_t) \succeq -\sqrt{\varepsilon}\mathbf{I}.$$

If the goal is simply $\|\nabla f(x_t)\| \leq \varepsilon$, then $T = \widetilde{O}\big(\left(\alpha^2 + \frac{1}{m}\right)\frac{(f(x_0) - \min f(x))}{\varepsilon^4}\big)$ iterations suffice. Here, the $\widetilde{O}$ notation serves to hide logarithmic factors for readability. We spell out these factors in the detailed analysis.

- When $\alpha < 1/\sqrt{m}$, our sample complexity ($= mT$) matches the best known result in the non-Byzantine case (Jin et al., 2019) without additional assumptions, and enjoys linear parallel speed-up: with $m$ workers of which $< \sqrt{m}$ are Byzantine, the parallel speedup is $\widetilde{\Omega}(m)$.[1]
- For $\alpha \in [1/\sqrt{m}, 1/2)$, our parallel time complexity is $\widetilde{O}(\alpha^2)$ times that needed when no parallelism is used. This still gives parallel speedup. This $\alpha^2$ factor appears in convex Byzantine distributed optimization, where it is tight (Yin et al., 2018; Alistarh et al., 2018).
- The Lipschitz smoothness and second-order smoothness assumptions are the minimal assumptions needed to derive convergence rates for finding second-order critical points (Jin et al., 2019).

**Comparison with prior bounds.** The closest known bounds are by Yin et al. (2019), who derived three gradient descent-type of algorithms (based on median, mean, and iterative filtering) to find a weaker type of approximate local minima. Since it relies on full gradients, their algorithm is arguably less practical, and their time complexities are generally higher than ours (see Section 2.1).

Other prior works consider a weaker goal: to find approximate stationary points $\|\nabla f(x)\| \leq \varepsilon$ only: Bulusu et al. (2020) additionally assumed there is a guaranteed good (i.e. non-Byzantine) worker known by the master, Xie et al. (2018b) gave a practical algorithm when the Byzantine attackers have no information about the loss function or its gradient, Yang et al. (2019); Xie et al. (2018a); Blanchard et al. (2017) derived eventual convergence without an explicit complexity bound, and the non-convex result obtained in Yin et al. (2018) is subsumed by Yin et al. (2019), discussed above.

**Our algorithm and techniques.** The structure of our algorithm is deceptively simple. The master node keeps track of the *sum of gradients* produced by *each* worker across time. It labels *(allegedly) good* workers as those whose sum of gradients "concentrate" well with respect to a surrogate of the median vector, and labels *bad* workers otherwise. Once a worker is labelled bad, it is removed from consideration forever. The master then performs the *vanilla SGD*, by moving in the negative direction of the average gradients produced by those workers currently labelled as good.

We call our algorithm `SafeguardSGD`, since it behaves like having a safe guard to filter away bad workers. Its processing overhead at the master is $O(md)$, negligible compared to standard SGD.

As the astute reader may have guessed, the key non-trivial technical ingredient is to identify the right quantity to check for concentration, and make it compatible with the task of non-convex optimization. In particular, we manage to construct such quantities so that (1) good non-Byzantine workers never get mislabelled as bad ones; (2) Byzantine workers may be labelled as good ones (which is inevitable) but when they do, the convergence rates are not impacted significantly; and (3) the notion does not require additional assumptions or running time overhead. To achieve these goals, we leverage a new concentration lemma that may be of independent interest (see Section 3).

The idea of using concentration (for each worker across time) to filter out Byzantine machines traces back to the convex setting (Alistarh et al., 2018). However, the quantities used in (Alistarh et al., 2018) to check for concentration are necessarily different from this paper, and our analysis is completely new, as deriving non-convex rates is known to be much more delicate and challenging. Recently, Bulusu et al. (2020) used similar concentration filters to Alistarh et al. (2018) in the non-convex setting, but under stronger assumptions, and for the simpler task of finding stationary points.

Many other algorithms *do not* rely on concentration filters. In each iteration, they ask each worker to compute *a batch* of stochastic gradients, and then use coordinate-wise median or mean over the

[1] By *parallel speedup* we mean the reduction in wall-clock time due to sampling gradients in parallel among the $m$ nodes. In each time step, the algorithm generates $m$ new gradients, although some may be corrupted.

batch average (e.g. Yin et al. (2018; 2019); Yang et al. (2019)) or iterative filtering (e.g. Su and Xu (2018); Yin et al. (2019)) by the master to derive a "robust mean." These works fundamentally rely on each iteration to calculate an *almost precise* full gradient, so that they can apply a surrogate of *full gradient descent*. Such algorithms can introduce higher sample and time complexities (see Section 2), are less practical than *stochastic gradient* schemes, require additional restrictions on the resilience factor $\alpha$, e.g. $\alpha < 1/4$ (Su and Xu, 2018), and, critically, have been shown to be vulnerable to recent attacks (Baruch et al., 2019; Xie et al., 2020).

**Attack resilience and experimental validation.** There is a growing literature on customized attacks against Byzantine-resilient algorithms, showing that many defenses can be entirely circumvented in real-world scenarios (Baruch et al., 2019; Xie et al., 2020). Our algorithm is provably correct against these attacks, a fact we also validate experimentally. Specifically, we implemented `SafeguardSGD` to examine its practical performance against a range of prior works (Xie et al., 2018b; Blanchard et al., 2017; Chen et al., 2017; Yin et al., 2018; 2019), and against recent attacks on the distributed task of *training deep neural networks*. Our experiments show that `SafeguardSGD` generally outperforms previous methods in convergence speed and final accuracy, sometimes by a wide accuracy margin. This is true not only against known Byzantine attacks, but also against attack variants we fine-crafted to specifically slow down our algorithm. `SafeguardSGD` provides two desirable properties: (1) `SafeguardSGD` quickly detects nodes which significantly misbehave; (2) the impact of the remaining Byzantine workers (with undetectable deviations) is limited in practice.

## 2 Statement of Our Theoretical Result

We denote by $\|\cdot\|$ the Euclidean norm and $[n] := \{1, 2, \ldots, n\}$. Given symmetric matrices $\mathbf{A}, \mathbf{B}$, we let $\|\mathbf{A}\|_2$ denote the spectral norm of $\mathbf{A}$. We use $\succeq$ to denote Loewner ordering, i.e. $\mathbf{A} \succeq \mathbf{B}$ if $\mathbf{A} - \mathbf{B}$ is positive semi-definite. We denote by $\lambda_{\min}(\mathbf{A})$ the minimum eigenvalue of matrix $\mathbf{A}$.

We consider arbitrary $d$-dimensional non-convex functions $f \colon \mathbb{R}^d \to \mathbb{R}$ satisfying the following:

- $f(x)$ is $L$-Lipschitz smooth: meaning $\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|$ for any $x, y \in \mathbb{R}^d$;
- $f(x)$ is $L_2$-second-order smooth: $\|\nabla^2 f(x) - \nabla^2 f(y)\|_2 \le L_2 \cdot \|x - y\|$ for any $x, y \in \mathbb{R}^d$;

For notational simplicity of the proofs, we assume $L = L_2 = \mathcal{V} = 1$.[2] Note that we have also assumed the domain of $f$ is the entire space $\mathbb{R}^d$. If instead there is a compact domain $\mathcal{X} \subset \mathbb{R}^d$, then one can use projected SGD and re-derive similar results of this paper. We choose to present our result in the simplest setting to convey our main ideas.

**Byzantine non-convex stochastic distributed optimization.** We let $m$ be the number of worker machines and assume at most an $\alpha$ fraction of them are Byzantine for $\alpha \in \left[0, \frac{1}{2}\right)$. We denote by $\mathsf{good} \subseteq [m]$ the set of good (i.e. non-Byzantine) machines, and the algorithm does not know $\mathsf{good}$.

**Assumption 2.1.** *In each iteration $t$, the algorithm (on the master) is allowed to specify a point $x_t$ and query $m$ machines. Each machine $i \in [m]$ gives back a vector $\nabla_{t,i} \in \mathbb{R}^d$ satisfying*

- *If $i \in \mathsf{good}$, the stochastic gradient $\nabla_{t,i}$ satisfies $\mathbb{E}[\nabla_{t,i}] = \nabla f(x_t)$ and $\|\nabla f(x_t) - \nabla_{t,i}\| \le \mathcal{V}$.*
- *If $i \in [m] \setminus \mathsf{good}$, then $\nabla_{t,i}$ can be arbitrary (w.l.o.g. we assume $\|\nabla f(x_t) - \nabla_{t,i}\| \le \mathcal{V}$).*[3]

*Remark* 2.2. For each $t$ and $i \notin \mathsf{good}$, the vector $\nabla_{t,i}$ can be adversarially chosen and may depend on $\{\nabla_{t',i}\}_{t' \le t, i \in [m]}$. In particular, the Byzantine machines can even collude during an iteration.

### 2.1 Our Algorithm and Theorem

Our algorithm is based on arguably the simplest possible method for achieving this goal, *(perturbed) stochastic gradient descent (SGD)* (Ge et al., 2015). Our techniques more broadly apply to more complicated methods (e.g. at least to Allen-Zhu (2018a;b)), but we choose to analyze the simplest variant of SGD, since it is the most widely applied method in modern non-convex machine learning.

As illustrated in Algorithm 1, in each iteration $t = 0, 1, \ldots, T-1$, we maintain a set of (allegedly)

[2] In the literature of convergence analysis for non-convex optimization, the final complexity bounds *naturally* and *polynomially* depend on these parameters $L, L_2, \mathcal{V}$, and the way the dependence goes is typically unique (Allen-Zhu, 2018a;b; Fang et al., 2018; Jin et al., 2019). This is why it suffices to ignore their appearance and only compare the polynomial dependence on $\varepsilon$ and $d$.

[3] This requirement $\|\nabla f(x_t) - \nabla_{t,i}\| \le \mathcal{V}$ is "without loss of generality" because it is trivial for the algorithm to catch bad machines if they output $\nabla_{t,i}$ that are more than $2\mathcal{V}$ away from the majorities.

**Algorithm 1** `SafeguardSGD`: perturbed SGD with double safe guard

**Input:** point $x_0 \in \mathbb{R}^d$, rate $\eta > 0$, lengths $T \geq T_1 \geq T_0 \geq 1$, threshold $\mathfrak{T}_1 > \mathfrak{T}_0 > 0$;

$\mathsf{good}_0 \leftarrow [m]$;
**for** $t \leftarrow 0$ **to** $T-1$ **do**
  $last_1 \leftarrow \max\{t_1 \in [t] \colon t_1 \text{ is a multiple of } T_1\}$;
  $last_0 \leftarrow \max\{t_0 \in [t] \colon t_0 \text{ is a multiple of } T_0\}$
  **for each** $i \in \mathsf{good}_t$ **do**
    receive $\nabla_{t,i} \in \mathbb{R}^d$ from machine $i$;
    $A_i \leftarrow \sum_{k=last_1}^{t} \frac{\nabla_{k,i}}{|\mathsf{good}_k|}$ and $B_i \leftarrow \sum_{k=last_0}^{t} \frac{\nabla_{k,i}}{|\mathsf{good}_k|}$;
  $A_{\mathsf{med}} \leftarrow A_i$ where $i \in \mathsf{good}_t$ is any machine s.t. $\left|\{j \in \mathsf{good}_t \colon \|A_j - A_i\| \leq \mathfrak{T}_1\}\right| > m/2$.
  $B_{\mathsf{med}} \leftarrow B_i$ where $i \in \mathsf{good}_t$ is any machine s.t. $\left|\{j \in \mathsf{good}_t \colon \|B_j - B_i\| \leq \mathfrak{T}_0\}\right| > m/2$.
  $\mathsf{good}_{t+1} \leftarrow \left\{i \in \mathsf{good}_t \colon \|A_i - A_{\mathsf{med}}\| \leq 2\mathfrak{T}_1 \bigwedge \|B_i - B_{\mathsf{med}}\| \leq 2\mathfrak{T}_0\right\}$ ;
  $x_{t+1} = x_t - \eta\left(\xi_t + \frac{1}{|\mathsf{good}_t|}\sum_{i \in \mathsf{good}_t} \nabla_{t,i}\right)$; $\diamond$ *Gaussian noise* $\xi_t \sim \mathcal{N}(0, \nu^2 \mathbf{I})$

good machines $\mathsf{good}_t \subseteq [m]$. We begin with $\mathsf{good}_0 = [m]$ and start to detect malicious machines and remove them from the set. We choose a learning rate $\eta > 0$, and perform the SGD update

$$x_{t+1} = x_t + \xi_t - \eta \frac{1}{|\mathsf{good}_t|} \sum_{i \in \mathsf{good}_t} \nabla_{t,i}$$

where $\xi_t \sim \mathcal{N}(0, \nu^2 \mathbf{I})$ is a random Gaussian perturbation that we introduce.

For each machine $i \in [m]$, we keep track of the history of its stochastic gradients up to *two windows*. Namely, $A_i \leftarrow \sum_{k=last_1}^{t} \frac{\nabla_{k,i}}{|\mathsf{good}_k|}$ and $B_i \leftarrow \sum_{k=last_0}^{t} \frac{\nabla_{k,i}}{|\mathsf{good}_k|}$, for windows sizes $T_0 \leq T_1 \leq T$. We compare among remaining machines in $\mathsf{good}_t$, and kick out those ones whose $A_i$ or $B_i$ deviate "more than usual" to construct $\mathsf{good}_{t+1}$. Conceptually, we view these two as *safe guards*.

Our theory makes sure that, when the "window sizes" and the thresholds for "more than usual" are defined properly, then $\mathsf{good}_t$ shall always include $\mathsf{good}$, and the algorithm shall proceed to find approximate local minima. Formally, we have (letting the $\widetilde{O}$ notion to hide polylogarithmic factors)

**Theorem 2.3.** *Let $C_3 = \alpha^2 + \frac{1}{m}$. Suppose we choose $\nu^2 = \widetilde{\Theta}(C_3)$, $\eta = \widetilde{\Theta}(\frac{\varepsilon^2}{dC_3})$, $T_0 = \widetilde{\Theta}(\frac{1}{\eta})$, $T_1 = \widetilde{\Theta}(\frac{1}{\eta\sqrt{\varepsilon}})$, $\mathfrak{T}_0 = \widetilde{\Theta}(\sqrt{T_0})$, and $\mathfrak{T}_1 = \widetilde{\Theta}(\sqrt{T_1})$, then after*

$$T = \widetilde{O}\left(\frac{(f(x_0) - \min f(x))d}{\varepsilon^4}(\alpha^2 + \frac{1}{m})\right)$$

*iterations, with high probability, for at least constant fraction of the indices $t \in [T]$, they satisfy*

$$\|\nabla f(x_t)\| \leq \varepsilon \quad \textit{and} \quad \nabla^2 f(x_t) \succeq -\sqrt{\varepsilon}\mathbf{I} \ .$$

*Remark* 2.4. If one only wishes to achieve a *significantly simpler goal* — finding first-order critical points $\|\nabla f(x_t)\| \leq \varepsilon$ — the analysis becomes much easier (see Section 4). In particular, having one safe guard without perturbation (i.e. $\nu = 0$) suffices, and the iteration complexity reduces to $T = \widetilde{O}\big(\frac{f(x_0) - \min f(x)}{\varepsilon^4}(\alpha^2 + \frac{1}{m})\big)$. Bulusu et al. (2020) achieves this easier goal but requires an additional assumption: there is one guaranteed good worker known by the master.

**Our contribution.** We reiterate our theoretical contributions from three perspectives. 1) When $\alpha < 1/\sqrt{m}$, our algorithm requires $mT = \widetilde{O}\big(\frac{(f(x_0) - \min f(x))d}{\varepsilon^4}\big)$ stochastic gradient computations. This matches the best known result (Jin et al., 2019) under our minimal assumptions of the non-convex objective. (There exist other works in the stochastic setting that break the $\varepsilon^{-4}$ barrier and get rid of the dimension dependence $d$ under stronger assumptions.)[4]. 2) When $\alpha < 1/\sqrt{m}$, our algorithm enjoys linear parallel speed-up: the parallel time complexity reduces by a factor of $\Theta(m)$. When $\alpha \in [1/\sqrt{m}, 1/2)$, our parallel time complexity is $\widetilde{O}(\alpha^2)$ times that needed when no parallelism is used, still giving noticeable speedup. The $\alpha^2$ factor also appeared in convex Byzantine distributed optimization (and is known to be tight there) (Yin et al., 2018; Alistarh et al., 2018).

**Comparison to Yin et al. (2019).** Yin et al. (2019) derived three gradient descent-type algo-

[4]Works such as (Allen-Zhu, 2018a; Lei et al., 2017; Tripuraneni et al., 2017; Allen-Zhu, 2018b; Fang et al., 2018; Nguyen et al., 2017) require $f(x) = \mathbb{E}_{s \sim \mathcal{D}}[f_s(x)]$ where each $f_s(x)$ is second-order smooth and/or Lipschitz smooth. This requirement may be too strong for certain practical applications.

rithms to find points with a weaker (and less standard) guarantee: $\|\nabla f(x)\| \leq \varepsilon$ and $\nabla^2 f(x) \succeq -(\varepsilon^2 d)^{1/5}\mathbf{I}$. Despite practical differences (namely, gradient descent may be less favorable comparing to stochastic gradient descent especially in deep learning applications), the parallel time complexities derived from their result are also generally larger than ours.

Their paper focuses on bounding the number of sampled stochastic functions, as opposed to the number of stochastic gradient evaluations like we do in this paper. When translated to our language, each of the workers in their setting needs to evaluate $T$ stochastic gradients, where (1) $T = \widetilde{O}\big(\frac{\alpha^2 d}{\varepsilon^4} + \frac{d^2}{\varepsilon^4 m} + \frac{\sqrt{d}}{\varepsilon^3}\big)$ if using coordinate-wise median, (2) $T = \widetilde{O}\big(\frac{\alpha^2 d^2}{\varepsilon^4} + \frac{d^2}{\varepsilon^4 m}\big)$ if using trimmed mean, and (3) $T = \widetilde{O}\big(\frac{\alpha}{\varepsilon^4} + \frac{d}{\varepsilon^4 m}\big)$ if using iterative filtering. The complexities (1) and (2) are larger than ours (also with a weaker guarantee); the complexity (3) seems incomparable to ours, but when translating to the more standard $(\varepsilon, \sqrt{\varepsilon})$ guarantee, becomes $T = \widetilde{O}\big(\frac{\alpha d^2}{\varepsilon^5} + \frac{d^3}{\varepsilon^5 m}\big)$ so is also larger than ours. It is worth noting that (3) requires $\alpha < 1/4$ so cannot withstand half of the machines being Byzantine.

**Resilience against practical attacks.** Recall that our algorithm's filtering technique is based upon tracking $B_i$, the stochastic gradients of each machine $i$ *averaged over a window of* $T_0$ *iterations*. This is a departure from previous defenses, most of which are history-less, and enables us to be provably Byzantine-resilient against recent state-of-the-art attacks (Baruch et al., 2019; Xie et al., 2020).

Specifically, in Baruch et al. (2019), Byzantine workers collude to shift the gradient mean by a factor $\beta$ times the standard deviation of the (true stochastic) gradient, while staying within population variance. They noticed $\beta$ can be quite large in practice, since stochastic gradients tend to have large variance, especially in neural network training. Their attack can circumvent existing defenses because those defense algorithms are "historyless", while their attack is statistically indistinguishable from an honest execution in a single iteration. However, our algorithm can provably defend against this attack since it has *memory*: Byzantine workers following their strategy will progressively diverge from the (honest) "median" $B_{\mathsf{med}}$ (by an amount proportional to $\Omega(T)$ in $T$ iterations), and thus be marked as malicious by our algorithm (since our safeguard threshold is proportional to $O(\sqrt{T})$). Alternatively, if the Byzantine workers attempt to disrupt the mean while staying within our algorithm's thresholds, then we prove their influence on convergence must be negligible. In Xie et al. (2020), Byzantine workers collude to deviate in the negative direction of the gradient. Similarly to the previous attack, to avoid being caught by our algorithm, the maximum "magnitude" of this attack again has to stay within our thresholds. We implemented both attacks and showed our algorithm's robustness experimentally.

Finally, we note that prior "historyless" schemes, such as Krum or median-based schemes, could be thought of as providing stronger guarantees, as they in theory allow Byzantine nodes to change IDs during the computation: such schemes only require an upper bound on the number of Byzantine agents in each round. However, the attack of Baruch et al. (2019) essentially shows that all such schemes are vulnerable to variance attacks, and that such attacks are eminently plausible in practice. Thus, this suggests that the use of historical information, which requires that Byzantine nodes cannot change their IDs during the execution, may be necessary for Byzantine resilience.

## 3 Concentration of Measure for Sums of Partial Sums

We begin with a new concentration result which is central to our analysis.

**Lemma 3.1.** *Fix $p \in (0,1)$. Suppose $\xi_0, \ldots, \xi_{T-1} \sim \mathcal{N}(0,1)$ are i.i.d. and $\Delta_1, \ldots, \Delta_{T-1} \in \mathbb{R}$. Here, each $\Delta_t$ can depend on $\xi_0, \ldots, \xi_{t-1}$ but not on $\xi_t, \ldots, \xi_{T-1}$. Assume that this sequence satisfies $|\Delta_1 + \cdots + \Delta_t|^2 \leq \mathfrak{T}$ for every $t = 1, \ldots, T-1$. Then, with probability at least $1-p$,*

$$\left|\sum_{t=1}^{T-1} (\xi_0 + \cdots + \xi_{t-1}) \cdot \Delta_t\right| \leq O(\sqrt{T\mathfrak{T}\log(T/p)}) \ .$$

In our final convergence analysis of `SafeguardSGD`, we shall apply Lemma 3.1 by substituting each $\xi_t$ with the Gaussian noise and each $\Delta_t$ with the Byzantine attack in iteration $t$.

The proof of this result is in Appendix A. We can extend this to a similar concentration lemma for inner products over $d$-dimensional vectors.

**Lemma 3.2.** *Fix the dimension parameter $d \geq 1$. Suppose $\xi_0, \ldots, \xi_{T-1} \in \mathbb{R}^d$ are i.i.d. drawn from $\mathcal{N}(0, \mathbf{I})$, and that $\Delta_1, \ldots, \Delta_{T-1}$ are arbitrary vectors in $\mathbb{R}^d$. Here, each vector $\Delta_t$ with*

**Algorithm 2** Perturbed SGD with single safe guard (for analysis purpose only)

**Input:** point $w_0 \in \mathbb{R}^d$, set $\mathsf{good}_0 \supseteq \mathsf{good}$, rate $\eta > 0$, length $T \geq 1$, threshold $\mathfrak{T} > 0$;
1: **for** $t \leftarrow 0$ **to** $T-1$ **do**
2: **for each** $i \in \mathsf{good}_t$ **do**
3: receive $\nabla_{t,i} \in \mathbb{R}^d$ from machine $i$;
4: $B_i \leftarrow \sum_{k=0}^{t} \frac{\nabla_{k,i}}{|\mathsf{good}_k|}$;
5: $B_{\mathsf{med}} \leftarrow B_i$ where $i \in \mathsf{good}_t$ is any machine s.t. $\left|\{j \in \mathsf{good}_t \colon \|B_j - B_i\| \leq \mathfrak{T}\}\right| > m/2$.
6: $\mathsf{good}_{t+1} \leftarrow \left\{i \in \mathsf{good}_t \colon \|B_i - B_{\mathsf{med}}\| \leq 2\mathfrak{T}\right\}$;
7: $w_{t+1} = w_t - \eta\left(\xi_t + \frac{1}{|\mathsf{good}_t|}\sum_{i\in\mathsf{good}_t} \nabla_{t,i}\right)$; ◇ *Gaussian noise* $\xi_t \sim \mathcal{N}(0, \nu^2\mathbf{I})$

*$t = 1, \ldots, T-1$ can depend on $\xi_0, \ldots, \xi_{t-1}$ but not on $\xi_t, \ldots, \xi_{T-1}$. Suppose that these vectors satisfy $\|\Delta_1 + \cdots + \Delta_t\|^2 \leq \mathfrak{T}$ for every $t = 1, \ldots, T-1$. Then, with probability at least $1-p$,*

$$\left|\sum_{t=1}^{T-1} \langle \xi_0 + \cdots + \xi_{t-1}, \Delta_t \rangle\right| \leq O(\sqrt{dT\mathfrak{T}\log(T/p)}) \ .$$

## 4 WARMUP: SINGLE SAFE GUARD

As a warmup, let us first analyze the behavior of perturbed SGD with a single safe guard. Consider Algorithm 2, where we start with a point $w_0$, a set $\mathsf{good}_0 \supseteq \mathsf{good}$, and perform $T$ steps of perturbed SGD. (We use the $w_t$ sequence instead of the $x_t$ sequence to emphasize that we are in Algorithm 2.)

**Definition 4.1.** *We make the following definition to simplify notations: let $\Xi_t := \sigma_t + \Delta_t$ where*

- $\sigma_t := \frac{1}{|\mathsf{good}_t|}\sum_{i\in\mathsf{good}}\left(\nabla_{t,i} - \nabla f(w_t)\right)$
- $\Delta_t := \frac{1}{|\mathsf{good}_t|}\sum_{i\in\mathsf{good}_t\setminus\mathsf{good}}\left(\nabla_{t,i} - \nabla f(w_t)\right)$

*Therefore, we can re-write the SGD update as $w_{t+1} = w_t - \eta(\nabla f(w_t) + \xi_t + \Xi_t)$ .*

The following lemma is fairly immediate to prove:

**Lemma 4.2** (single safe guard)**.** *In Algorithm 2, suppose we choose $\mathfrak{T} = 8\sqrt{T\log(16mT/p)}$. Then, with probability at least $1 - p/4$, for every $t = 0, \ldots, T-1$,*

- $\mathsf{good}_t \supseteq \mathsf{good}$.
- $\|\sigma_t\|^2 \leq O(\frac{\log(T/p)}{m})$ *and* $\|\sigma_0 + \cdots + \sigma_{t-1}\|^2 \leq O(\frac{T\log(T/p)}{m})$
- $\|\Delta_t\|^2 \leq \alpha^2$ *and* $\|\Delta_0 + \cdots + \Delta_{t-1}\|^2 \leq O(\alpha^2 T\log(mT/p))$
- $|\langle \nabla f(w_t), \xi_t\rangle| \leq \|\nabla f(w_t)\| \cdot O(\nu\sqrt{\log(T/p)})$,
- $\|\xi_t\|^2 \leq O(\nu^2 d\log(T/p))$, $\|\xi_0 + \cdots + \xi_{t-1}\|^2 \leq O(\nu^2 dT\log(T/p))$

*We call this probabilistic event $\mathsf{Event}_T^{\mathsf{single}}(w_0)$ and $\Pr[\mathsf{Event}_T^{\mathsf{single}}(w_0)] \geq 1 - p/4$.*

(The third property above is ensured by our choice of $\mathfrak{T}$ and the use of safe guard, and the rest of the properties follow from simple martingale concentration arguments. Details are in Appendix B.1.)

**Core Technical Lemma 1: Objective Decrease.** Our first main technical lemma is the following:

**Lemma 4.3.** *Suppose we choose $\mathfrak{T}$ as in Lemma 4.2. Denote by $C_1 = \log(T/p)$ and $C_2 = \alpha^2\log\frac{mT}{p} + \frac{\log(T/p)}{m}$. Suppose $\eta \leq 0.01\min\{1, \frac{1}{C_2}\}$, $T = \frac{1}{100\eta(1+\sqrt{C_2})}$ and we start from $w_0$ and apply Algorithm 2. Under event $\mathsf{Event}_T^{\mathsf{single}}(w_0)$, it satisfies*

$$f(w_0) - f(w_T) \geq 0.7\eta\sum_{t=0}^{T-1}\left(\|\nabla f(w_t)\|^2 - \eta\cdot O(C_2 + (C_2)^{1.5}) - O(C_1\nu^2\eta(d + \sqrt{C_2}))\right)$$

Lemma 4.3 says after $T \approx \frac{1}{\eta}$ steps of perturbed SGD, the objective value decreases by, up to some small additive error and *up to logarithmic factors*, $f(w_0) - f(w_T) \geq 0.7\eta\sum_{t=0}^{T-1}(\|\nabla f(w_t)\|^2 - \eta C_2)$. This immediately implies, if we choose $\eta \approx \frac{\varepsilon^2}{C_2}$, then by repeating this analysis for $O(\frac{C_2}{\varepsilon^4}) = O(\frac{\alpha^2 + 1/m}{\varepsilon^4})$ iterations, we can find approximate critical point $x$ with $\|\nabla f(x)\| \leq \varepsilon$.

*Proof sketch of Lemma 4.3.* The full proof is in Appendix B.2 but we illustrate the main idea and

difficulties below. After simple manipulations, it is not hard to derive that

$$f(w_0) - f(w_T) \gtrapprox 0.9\eta \sum_{t=0}^{T-1} \left( \|\nabla f(w_t)\|^2 - \eta \right) + \underbrace{\eta \sum_{t=0}^{T-1} \langle \nabla f(w_t), \Xi_t \rangle}_{\text{remainder terms}}$$

where recall that $\Xi_t = \sigma_t + \Delta_t$. When there are no Byzantine machines, we have $\mathbb{E}[\Xi_t] = \mathbb{E}[\sigma_t] = 0$ so the remainder terms must be small by martingale concentration. Therefore, the main technical difficulty arises to deal with those Byzantine machines, who can adversarially design their $\nabla_t$ (even by collusion) so as to negatively correlate with $\nabla f(w_t)$ to "maximally destroy" the above inequality.

Our main idea is to use second-order smoothness to write $\nabla f(w_t) \approx \nabla f(w_0) + \nabla^2 f(w_0) \cdot (w_t - w_0)$. To illustrate our idea, let us ignore the constant vector and assume that the Hessian is the identity: that is, imagine as if $\nabla f(w_t) \approx w_t - w_0$. Using $w_t - w_0 = -\sum_{k<t} \Xi_t + \xi_t$, we immediately have

$$-\langle \nabla f(w_t), \Xi_t \rangle \approx -\langle w_t - w_0, \Xi_t \rangle = \textstyle\sum_{k<t} \langle \Xi_k, \Xi_t \rangle + \sum_{k<t} \langle \xi_k, \Xi_t \rangle \tag{4.1}$$

For the first partial sum $\langle \sum_{k<t} \Xi_k, \Xi_t \rangle$ in (4.1), it is easy to bound its magnitude using our safeguard. Indeed, we have $\left| \sum_t \langle \sum_{k<t} \Xi_k, \Xi_t \rangle \right| \leq \| \sum_t \Xi_t \|^2 + \sum_t \|\Xi_t\|^2$ so we can apply Lemma 4.2. For the second partial sum $\sum_t \sum_{k<t} \langle \xi_k, \Xi_t \rangle$, we can apply Lemma 3.2. □

**Core Technical Lemma 2: Randomness Coupling.** Our next technical lemma studies that, if run Algorithm 2 from a point $w_0$ so that the Hessian $\nabla^2 f(w_0)$ has a eigenvalue which is less than $-\delta$ (think of $w_0$ as a saddle point), then with good probability, after sufficiently many iterations, the sequence $w_1, w_2, \ldots, w_T$ shall *escape* from $w_0$ to distance at least $R$ for some parameter $R \approx \delta$. To prove this, motivated by Jin et al. (2017), we study two executions of Algorithm 2 where their randomness are coupled. We then argue that at least one of them has to escape from $w_0$. For any vector $v$, let $[v]_i$ denote the $i$-th coordinate of $v$.

**Lemma 4.4.** *Suppose we choose $\mathfrak{T}$ as in Lemma 4.2 and $C_1, C_2$ as in Lemma 4.3. Suppose $w_0 \in \mathbb{R}^d$ satisfies $\lambda_{\min}(\nabla^2 f(w_0)) = -\delta$ for some $\delta \geq 0$. Without loss of generality let $\mathbf{e}_1$ be the eigenvector of $\nabla^2 f(w_0)$ with smallest eigenvalue. Consider now two executions of Algorithm 2, both starting from $w_0^{\mathsf{a}} = w_0^{\mathsf{b}} = w_0$, and suppose their randomness $\{\xi_t^{\mathsf{a}}\}_t$ and $\{\xi_t^{\mathsf{b}}\}_t$ are coupled so that $[\xi_t^{\mathsf{a}}]_1 = -[\xi_t^{\mathsf{b}}]_1$ but $[\xi_t^{\mathsf{a}}]_i = [\xi_t^{\mathsf{b}}]_i$ for $i > 1$. In words, the randomness is the same orthogonal to $\mathbf{e}_1$, but along $\mathbf{e}_1$, the two have opposite signs. Now, suppose we perform $T = \Theta(\frac{1}{\eta\delta} \log \frac{R^2\delta}{\eta\nu^2})$ steps of perturbed SGD from $w_0^{\mathsf{a}}, w_0^{\mathsf{b}}$ respectively using Algorithm 2. Suppose*

$$R \leq O\big(\tfrac{\delta}{\sqrt{C_1} \log(R^2\delta/\eta\nu^2)}\big) \quad \textit{and} \quad \nu^2 \geq \Omega\big(C_2 \log \tfrac{R^2\delta}{\eta\nu}\big) \ .$$

*Then, under events $\mathsf{Event}_T^{\mathsf{single}}(w_0^{\mathsf{a}})$ and $\mathsf{Event}_T^{\mathsf{single}}(w_0^{\mathsf{b}})$, with probability at least 0.98, either $\|w_t^{\mathsf{a}} - w_0\| > R$ or $\|w_t^{\mathsf{b}} - w_0\| > R$ for some $t \in [T]$.*

Proof details in Appendix B.3. The main proof difficulty is to analyze a noisy version of the power method, where the noise comes from (1) Gaussian perturbation (which is the good noise), (2) stochastic gradients (which has zero mean), and (3) Byzantine workers (which can be adversarial).

**From Warmup to Final Theorem with Double Safe Guards.** At a high level, Lemma 4.3 ensures that if we keep encountering points with large gradient $\|\nabla f(w_t)\|$, then the objective value should sufficiently decrease; in contrast, Lemma 4.4 says that if we keep encountering points with negative Hessian directions (i.e., $\lambda_{\min}(\nabla^2 f(w_t)) < -\delta$), then the points must move a lot (i.e., by more than $R$ in $T$ iterations, which can also lead to sufficient objective decrease, see Lemma C.4). Therefore, at a high level, when the two lemmas are combined, they should tell that we *must not* encounter points with $\|\nabla f(x)\|$ being large, or $\lambda_{\min}(\nabla^2 f(x))$ being very negative, for too many iterations. Therefore, the algorithm can find approximate local minima. The reason we eventually need *two safe guards*, is because the number of rounds $T$ for Lemma 4.3 and Lemma 4.4 differ by a factor. Thus, we need two safe guards with different window sizes to ensure that the two lemmas simultaneously hold. We encourage the reader to examine the full analysis in Appendix C.

## 5 Experimental Validation

We evaluate the convergence of `SafeguardSGD` to examine its practical performance against prior works (Xie et al., 2018b; Blanchard et al., 2017; Chen et al., 2017; Yin et al., 2018; 2019). We perform the non-convex task of training a deep residual network (ResNet-20) (He et al., 2016) on

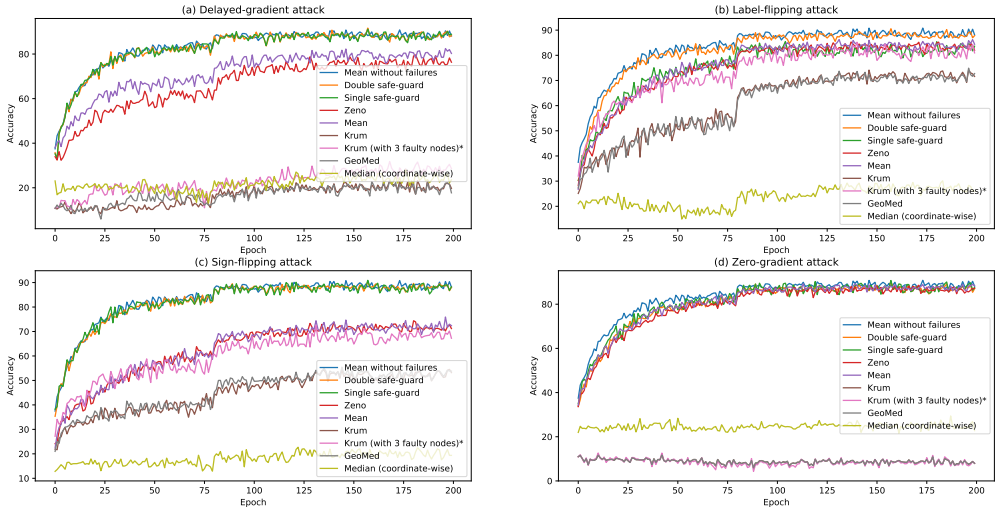


Figure 1: Convergence comparison (CIFAR-10 test accuracy) under different attacks. Figures for the Safeguard and Variance attacks are deferred to the Appendix due to space limitations.

| **Method** | **Test Accuracy / Attack Type** | | | | | |
|---|---|---|---|---|---|---|
| | Delayed-gradient | Label-flipping | Sign-flipping | Zero-gradient | Safeguard Attack (rescaling-factor = 0.4) | Variance Attack (stdev factor = 0.3) |
| Ideal Baseline | 90.8 | 90.8 | 90.8 | 90.8 | 90.8 | 90.8 |
| Double Safe-guard | 91.7 | 90.6 | 90.5 | 90.3 | 88.3 | 90.8 |
| Single Safe-guard | 91.1 | 85.5 | 90.9 | 90.6 | 87.5 | 91.0 |
| Best Other Method | 79.6 (Zeno) | 85.3 (Zeno) | 74.4 (Zeno) | 88.6 (Zeno) | 87.7 (Zeno) | 29.0 (Zeno) |

Table 1: Test accuracy comparison under different attacks. For full results see Table 3 in the Appendix.

the CIFAR-10 dataset (Krizhevsky et al., 2014). A full experimental report is given in the Appendix.

We instantiate $m = 10$ workers and one master executing data-parallel SGD for 200 passes (i.e. epochs) over the training dataset.[5] The results for higher number of workers are similar.

The *ideal baseline* is gradient mean without attacks, and we compare against *(Naive) Mean*, *Geometric Median* (Chen et al., 2017), *Coordinate-wise Median* (Yin et al., 2018; 2019), *Krum* (Blanchard et al., 2017), and *Zeno* (Xie et al., 2018b). Overall, our experimental setup is very similar to Zeno (Xie et al., 2018b) but with additional attacks. Results are averaged over three repetitions.

We implemented the approach of Yang et al. (2019), but found it very sensitive to hyper-parameter values and were not able to make it converge across all attacks even after significant tuning of its $\gamma$ parameter. We also implemented the convex algorithm of Alistarh et al. (2018), and executed it in our setting. However, we found that their algorithm can be successfully prevented from converging in the non-convex setting. Specifically, there exists a simple attack, described in Appendix D.5 which causes their algorithm to either (1) mislabel most good workers as Byzantine, or (2) completely diverge, or (3) eventually converge to a poor accuracy solution. This behaviour is not surprising, since their algorithm is designed for, and only guaranteed to work in, the convex setting. Please see Appendix D.5 for details.

To make the comparison stronger, when implementing `SafeguardSGD`, we have chosen fixed window sizes $T_0 = 1$ epoch and $T_1 = 6$ epochs across all experiments, and adopted an automated process to select $\mathfrak{T}_0, \mathfrak{T}_1$ (by pre-running the experiment for 20 epochs). We have also implemented a single safeguard variant of `SafeguardSGD` with window size $T = 3$ epochs.

**Attackers.** We set $\alpha = 0.4$ so there are 4 Byzantine workers. (This exceeds the fault-tolerance of Krum, and so we also tested Krum with only 3 Byzantine workers.)

- SIGN-FLIPPING ATTACK: each Byzantine worker sends the negative gradient to the master.

[5] As common in the literature, we omit the theoretical noise component to the gradient, and instead rely on the natural noise in the training process.

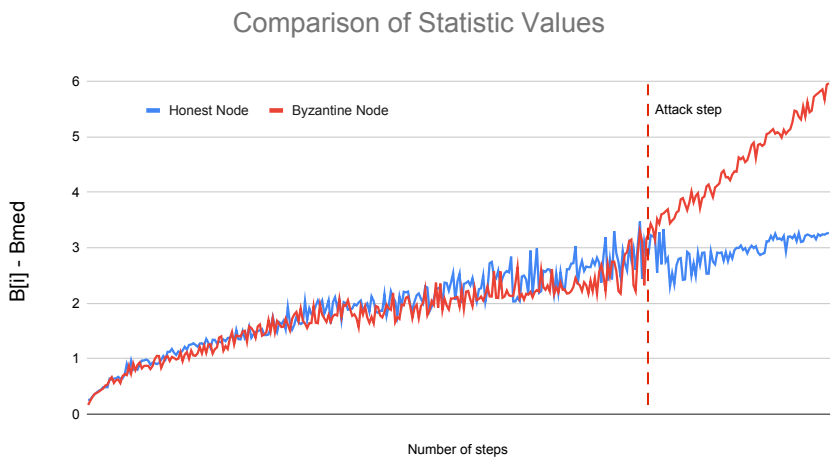


Table 2: Comparison of the $\|B_i - B_{\text{med}}\|$ statistic between a good node (blue), and a bad node (red) which pretends to be honest for the first half-epoch, and then starts applying the variance attack. Notice the sharp increase in the statistic at the point where the attack starts, marked by the dotted line.

- ZERO-GRADIENT ATTACK: each Byzantine worker sends zero gradient to the master.
- LABEL-FLIPPING ATTACK: each Byzantine worker computes its gradient based on the cross-entropy loss with flipped labels: for CIFAR-10, label $\ell \in \{0, ..., 9\}$ is flipped to $9 - \ell$.
- DELAYED-GRADIENT ATTACK: each Byzantine worker sends an *old* gradient to master. In our experiments, the delay is of $D = 1000$ iterations.
- SAFEGUARD ATTACK: Byzantine workers send negative but re-scaled gradient to the master. The re-scale factor $0.4$ is chosen to avoid triggering the safe-guard conditions at the master (we also show results for re-scale factor $0.5$ in the Appendix). This attack is an instantiation of the inner-product attack (Xie et al., 2020), customized specifically to maximally affect our algorithm.
- VARIANCE ATTACK (Baruch et al., 2019): Byzantine workers measure the mean and the standard-deviation of gradients at each round, and collude to move the mean by the largest value which still operates within population variance. (For our parameter settings, this is $0.3$ times the standard deviation. We discuss results for additional parameter values in the Appendix.)

**Discussion of results.** Figure 1 compares the convergence curves, while Table 1 compares the best test accuracy. We encourage the reader to examine the Appendix for full convergence results across all methods, as well as training accuracy and loss graphs. `SafeguardSGD` generally outperforms the previous methods in test accuracy and convergence, and closely tracks the performance of the ideal baseline, across all attacks. The test accuracy difference can be $> 10\%$ between our algorithm and the best prior work (see delayed gradient, sign-flipping, and variance attacks). `SafeguardSGD` slightly outperforms all other algorithms even for the customized safeguard attacks, which were designed to maximally impact its performance. In most cases, the single-safeguard algorithm is close to double-safeguard, except for the label-flipping attack. We conclude that `SafeguardSGD` can be practical, and outperforms previous approaches.

**Analyzing a specific attack.** It is interesting to consider the inner workings of the algorithm in the context of a "delayed" attack, where the Byzantine nodes collude to execute an attack after a specific, given, point in the execution (in this case, the first epoch), rather than all starting their attack at the very beginning of the execution. Figure 2 presents the results from the perspective of the value of $\|B_i - B_{\text{med}}\|$ registered at the central server, for two nodes, an honest one, and a Byzantine one. The Byzantine node pretends to be honest for the first half of the epoch, and then colludes with the others to apply the variance attack. (The results are similar for other attacks and starting points.)

We observe the following. First, note that the value of $\|B_i - B_{\text{med}}\|$ increases for all nodes during the execution (at a rate of roughly $\sqrt{T}$). Second, once the attack starts, the statistic for the Byzantine node starts to grow roughly *linearly*, as in this case the node deviates by a constant times the standard deviation in every step. This will lead to fast detection of such nodes. Alternatively, the bad nodes may "spread out" their attack to avoid detection, but the analysis shows that they may not significantly influence the result in this case.

## REFERENCES


Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 4613–4623, 2018.

Zeyuan Allen-Zhu. Natasha 2: Faster Non-Convex Optimization Than SGD. In *NeurIPS*, 2018a. Full version available at `http://arxiv.org/abs/1708.08694`.

Zeyuan Allen-Zhu. How To Make the Gradients Small Stochastically. In *NeurIPS*, 2018b. Full version available at `http://arxiv.org/abs/1801.02982`.

Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. In *Advances in Neural Information Processing Systems*, pages 8635–8645, 2019.

Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NIPS*, pages 118–128, 2017.

Saikiran Bulusu, Prashant Khanduri, Pranay Sharma, and Pramod K Varshney. On distributed stochastic gradient descent for nonconvex functions in the presence of byzantines. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3137–3141. IEEE, 2020.

Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, 1999.

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

Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, pages 689–699, 2018.

Jiashi Feng, Huan Xu, and Shie Mannor. Distributed robust learning. *arXiv preprint arXiv:1409.5937*, 2014.

Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Proceedings of the 28th Annual Conference on Learning Theory*, COLT 2015, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1724–1732. JMLR. org, 2017.

Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M Kakade, and Michael I. Jordan. On nonconvex optimization for machine learning: Gradients, stochasticity, and saddle points. *arXiv preprint arXiv:1902.04811*, 2019.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The CIFAR-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

Lihua Lei, Cheng Ju, Jianbo Chen, and Michael I Jordan. Nonconvex Finite-Sum Optimization Via SCSG Methods. In *NIPS*, 2017.

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

Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2613–2621. JMLR. org, 2017.

Iosif Pinelis. Optimum bounds for the distributions of martingales in banach spaces. *The Annals of Probability*, pages 1679–1706, 1994.

Lili Su and Nitin H Vaidya. Fault-tolerant multi-agent optimization: optimal iterative distributed algorithms. In *PODC*, pages 425–434. ACM, 2016a.

Lili Su and Nitin H Vaidya. Defending non-bayesian learning against adversarial attacks. *ISDC*, 2016b.

Lili Su and Jiaming Xu. Securing distributed machine learning in high dimensions. *arXiv preprint arXiv:1804.10140*, 2018.

Nilesh Tripuraneni, Mitchell Stern, Chi Jin, Jeffrey Regier, and Michael I Jordan. Stochastic Cubic Regularization for Fast Nonconvex Optimization. *ArXiv e-prints*, abs/1711.02838, November 2017.

Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Generalized Byzantine-tolerant SGD. *arXiv preprint arXiv:1802.10116*, 2018a.

Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Zeno: Byzantine-suspicious stochastic gradient descent. *arXiv preprint arXiv:1805.10032*, 2018b.

Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Fall of empires: Breaking byzantine-tolerant sgd by inner

product manipulation. volume 115 of *Proceedings of Machine Learning Research*, pages 261–270, Tel Aviv, Israel, 22–25 Jul 2020. PMLR. URL http://proceedings.mlr.press/v115/xie20a.html.

Haibo Yang, Xin Zhang, Minghong Fang, and Jia Liu. Byzantine-resilient stochastic gradient descent for distributed learning: A lipschitz-inspired coordinate-wise median approach. *arXiv preprint arXiv:1909.04532*, 2019.

Dong Yin, Yudong Chen, Kanna Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018.

Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Defending against saddle point attack in byzantine-robust distributed learning. In *International Conference on Machine Learning*, pages 7074–7084, 2019.

# APPENDIX

## A MISSING PROOFS IN SECTION 3

**Lemma 3.1.** *Suppose $\xi_0, \dots, \xi_{T-1} \sim \mathcal{N}(0,1)$ are i.i.d. and $\Delta_1, \dots, \Delta_{T-1} \in \mathbb{R}$. Here, each $\Delta_t$ can depend on $\xi_0, \dots, \xi_{t-1}$ but not on $\xi_t, \dots, \xi_{T-1}$. Suppose $\Delta$ satisfies $|\Delta_1 + \cdots + \Delta_t|^2 \le \mathfrak{T}$ for every $t = 1, \dots, T-1$. Then, with probability at least $1-p$,*

$$\left| \sum_{t=1}^{T-1} (\xi_0 + \cdots + \xi_{t-1}) \cdot \Delta_t \right| \le O(\sqrt{T\mathfrak{T}\log(T/p)})$$

*Proof of Lemma 3.1.* Without loss of generality, we can assume $\Delta_1 + \cdots + \Delta_{T-1} = 0$. This is so because, if not, then we can add an additional term $\Delta_T = -(\Delta_1 + \cdots + \Delta_{T-1})$, but this affects the total sum by at most $O(\sqrt{T\mathfrak{T}\log(T/p)})$ with probability at most $1-p$. Also without loss of generality, we can assume $\xi_0 = 0$ because it affects the sum by at most $\sqrt{\mathfrak{T}\log(T/p)}$ with probability at most $1-p$.

Now, let us convert the vector $(\Delta_1, \dots, \Delta_{T-1})$ into a set of "weighted segments". Let $i$ be the minimal index so that $\Delta_i \neq 0$.

- If $\Delta_i > 0$, we let $j > i$ be the minimal index so that $\Delta_j < 0$, and we add a "negative segment" $[i,j]$ with weight $w = -\min\{|\Delta_i|, |\Delta_j|\}$. Then, we replace $\Delta_i$ with $\Delta_i + w$ and replace $\Delta_j$ with $\Delta_j - w$ and continue the conversion.
- If $\Delta_i < 0$, we let $j > i$ be the minimal index so that $\Delta_j > 0$, and we add a "positive segment" $[i,j]$ with weight $w = \min\{|\Delta_i|, |\Delta_j|\}$. Then, we replace $\Delta_i$ with $\Delta_i + w$ and replace $\Delta_j$ with $\Delta_j - w$ and continue the conversion.

This process ends up with a set $\mathcal{S}$ of at most $T$ segments, each denoted as $([i,j], w)$, representing a segment $[i,j]$ with weight $w$. It is clear that

$$\sum_{t=1}^{T-1} \langle \xi_1 + \cdots + \xi_{t-1}, \Delta_t \rangle = \sum_{([i,j],w) \in \mathcal{S}} w \cdot (\xi_i + \cdots + \xi_{j-1}) \ . \tag{A.1}$$

It is a simple exercise to verify that,

**Claim A.1.** *For every point $x \in \{1.5, 2.5, \dots, T-1.5\}$, $x$ can be covered by (1) either only positive segments or only negative segments, and (2) with total absolute weight at most $\sqrt{\mathfrak{T}}$.*

*Proof.* Indeed, suppose by way of contradiction $x$ is covered by both some positive segment $[i_1, j_1]$ and by some negative segment $[i_2, j_2]$. Suppose without loss of generality that $[i_1, j_1]$ is created *before* $[i_2, j_2]$ through the above construction. Then, before $[i_1, j_1]$ is added, we know $\Delta_{i_1} < 0$ and $\Delta_{j_1} > 0$. However, we also know $\Delta_{i_2} > 0$ (because otherwise $[i_2, j_2]$ wouldn't be picked as a negative segment in later rounds). This violates the definition of $j_1$, which is the minimal index $j > i_1$ satisfying $\Delta_j > 0$.

As for the total weight, suppose without loss of generality that all segments covering $x$ are positive, with total weight more than $\sqrt{\mathfrak{T}}$. This necessarily means $\Delta_1 + \cdots + \Delta_{x-0.5} < -\sqrt{\mathfrak{T}}$ and it contradicts with our assumption. □

In the remainder of the proof we wish to bound the summation in (A.1). For notation simplicity, we assume as if only positive segments exist in $\mathcal{S}$ (to deal also negative segments, the proof is analogous).

We make a definition of *segment chain*. We say $\mathcal{C} = (([i_1, j_1], w_1), \dots, ([i_k, j_k], w_k))$ is a segment chain of weight $w$ if:

- $i_1 < j_1 \le i_2 < j_2 \le \cdots \le i_k < j_k$ (non of the two chains overlap), and
- $w_1 = \cdots = w_k = w$ (all segments have weight $w$).

We also make a definition of *chain decomposition*. We say that a set $\mathcal{S}$ of segments can be decomposed into segment chains $\{\mathcal{C}_k\}_k$, if for each segment $([i,j], w) \in \mathcal{S}$, let $\mathcal{C}_1, \dots, \mathcal{C}_m$ be all the chains that contain segment $[i,j]$, with individual chain weights $w_1, \dots, w_m$, then $w_1 + \cdots + w_m = w$. When this happens, we write $\mathcal{S} = \sum_k \mathcal{C}_k$.

**Claim A.2.** *We can decompose $\mathcal{S} = \sum_k \mathcal{C}_k$ into at most $T$ chains with total weights at most $\sqrt{\mathfrak{T}}$.*

*Proof.* Recall from the previous claim that in $\mathcal{S}$, each point $x \in \{1.5, 2.5, \dots, T-1.5\}$ is covered by at most total absolute weight $\sqrt{\mathfrak{T}}$. We call this quantity the "cover number" of $\mathcal{S}$.

We shall recursively construct chains out of $\mathcal{S}$. Let the first chain $\mathcal{C}$ be constructed as follows. Pick the lowest index $i_1$ such that some segment $([i_1, j_1], w_1)$ is in $\mathcal{S}$. Then, pick the lowest index $i_2 \ge j_1$ such that some

segment $([i_2, j_2], w_2)$ is in $\mathcal{S}$, and so on. In the end, we arrive at a sequence of segments $[i_1, j_1], \ldots, [i_2, j_2]$, and let $w = \min\{w_1, \ldots, w_k\}$. This gives a segment chain

$$\mathcal{C} = ([i_1, j_1], w), \cdots, ([i_k, j_k], w))$$

and we can remove it from $\mathcal{S}$ (by setting each $w_q \leftarrow w_q - w$ or removing the segment if $w_q = w$). Let the remaining segments be $\mathcal{S}'$ so we have $\mathcal{S} = \mathcal{S}' + \mathcal{C}$. Thus, we proceed to construct the second chain from $\mathcal{S}'$ and so on. There are at most $T$ chains.

We claim that, each time after removing a segment chain of weight $w$ from $\mathcal{S}$, the "cover number" reduces by $w$.

Indeed, suppose the chain we construct is $\mathcal{C}$ with total weight $w$, and suppose $\mathcal{S} = \mathcal{S}' + \mathcal{C}$. Suppose the "cover number" of $\mathcal{S}$ is $cov$. Then, for every $x \in \{1.5, 2.5, \ldots, T - 1.5\}$, if $x$ is covered by $\mathcal{C}$ then obviously $x$ is covered in $\mathcal{S}'$ with a total weight at most $cov - w$. Otherwise, if $x$ is not covered by $\mathcal{C}$, then let $([i_1, j_1], w_1), \ldots, ([i_k, j_k], w_k)$ be all the segments in $\mathcal{S}$ that have covered $x$. We know that none of them have appeared in $\mathcal{C}$ as otherwise $x$ is covered by $\mathcal{C}$. Now, let $([i, j], w) \in \mathcal{C}$ be the segment in the chain with largest $j < x$. It must satisfy that $i_1, i_2, \ldots i_k < j$ as otherwise one of those $k$ segments would have been added to $\mathcal{C}$. This means, at point $x' = j - 0.5$, this point $x'$ is covered by not only $([i_1, j_1], w_1), \ldots, ([i_k, j_k], w_k)$, but also by $([i, j], w)$ in $\mathcal{S}$. By the definition of cover number, it satisfies $w_1 + \cdots + w_k + w \leq cov$. In other words, $w_1 + \cdots + w_k \leq cov - w$.

In sum, we have shown that after removing $\mathcal{C}$, the remaining set $\mathcal{S}'$ of segments have cover number at most $cov - w$. This means, when the decomposition of chains finishes, the total chain weights do not exceed the initial cover number, which is $\sqrt{\mathfrak{T}}$. □

Finally, we consider the partial sum with respect to each segment chain $\mathcal{C}$ from the above construction. That is,

$$\sum_{([i,j],w) \in \mathcal{C}} w \cdot (\xi_i + \cdots + \xi_{j-1})$$

Suppose the chain $\mathcal{C}$ contains segments $([i_1, j_1], w), \ldots, ([i_k, j_k], w)$, the we have

$$\sum_{([i,j],w) \in \mathcal{C}} w \cdot (\xi_i + \cdots + \xi_{j-1}) = w \cdot ((\xi_{i_1} + \cdots + \xi_{j_1 - 1}) + \cdots + (\xi_{i_k} + \cdots + \xi_{j_k - 1}))$$

Now, one can carefully verify that given our construction, $i_1 < j_1 \leq i_2 < j_2 \leq \cdots \leq i_k < j_k$ is a set of multiple stopping points. That is, $i_1$ only depends on the randomness of $\xi_1, \ldots, \xi_{i_1 - 1}$, $j_1$ only depends on the randomness of $\xi_1, \ldots, \xi_{j_1 - 1}$, and so on. By concentration on sum of Gaussian variables with stopping times, we have with probability at least $1 - p/T$,

$$\|((\xi_{i_1} + \cdots + \xi_{j_1 - 1}) + \cdots + (\xi_{i_k} + \cdots + \xi_{j_k - 1}))\| \leq O(\sqrt{T \log(T/p)})$$

Summing up over all the chains we finish the proof of Lemma 3.1. ■

**Lemma 3.2.** *Suppose $\xi_0, \ldots, \xi_{T-1} \in \mathbb{R}^d$ are i.i.d. drawn from $\mathcal{N}(0, \mathbf{I})$ are $\Delta_1, \ldots, \Delta_{T-1} \in \mathbb{R}^d$. Here, each $\Delta_t$ can depend on $\xi_0, \ldots, \xi_{t-1}$ but not on $\xi_t, \ldots, \xi_{T-1}$. Suppose $\Delta$ satisfies $\|\Delta_1 + \cdots + \Delta_t\|^2 \leq \mathfrak{T}$ for every $t = 1, \ldots, T - 1$. Then, with probability at least $1 - p$,*

$$\left| \sum_{t=1}^{T-1} \langle \xi_0 + \cdots + \xi_{t-1}, \Delta_t \rangle \right| \leq O(\sqrt{dT\mathfrak{T} \log(T/p)})$$

*Proof of Lemma 3.2.* The proof is analogous, but this time for each segment $[i, j]$, we also need to attach to it a color $c \in \{1, 2, \ldots, d\}$. Now, for each $x$, let $w_1, \ldots, w_d$ be the total weights of segments that cover $x$ with color $1, \ldots, d$ respectively, then we have $w_1^2 + \cdots w_d^2 \leq \mathfrak{T}$ from our assumption. However, by AM-GM, this means $|w_1| + \cdots + |w_d| \leq \sqrt{\mathfrak{T} d}$. Therefore, the rest of the proof proceeds by replacing $\sqrt{\mathfrak{T}}$ with $\sqrt{d\mathfrak{T}}$. □

# B Missing Proofs for Section 4

## B.1 Proof of Lemma 4.2

Recall the following, useful inequality.

**Lemma B.1** (Pinelis' 1994 inequality (Pinelis, 1994))**.** *Let $X_1, \ldots, X_T \in \mathbb{R}^d$ be a random process satisfying $\mathbb{E}[X_t | X_1, \ldots, X_{t-1}] = 0$ and $\|X_t\| \leq M$. Then, $\Pr\left[\|X_1 + \cdots + X_T\|^2 > 2\log(2/\delta) M^2 T\right] \leq \delta$.*

Lemma 4.2 is in fact a direct corollary of the following claim, whose proof is quite classical. Denote by $C = \log(16mT/p)$. Denote by

$$B_i^{(t)} := \frac{\nabla_{0,i}}{|\mathsf{good}_0|} + \cdots + \frac{\nabla_{t-1,i}}{|\mathsf{good}_{t-1}|} \quad \text{and} \quad B_\star^{(t)} := \frac{\nabla f(w_0)}{|\mathsf{good}_0|} + \cdots + \frac{\nabla f(w_{t-1})}{|\mathsf{good}_{t-1}|}.$$

Recall at iteration $t-1$, Algorithm 2 computes $\{B_1^{(t)}, \dots, B_m^{(t)}\}$ as well as some $B_{\mathsf{med}}^{(t)} = B_i^{(t)}$ where $i$ is any machine in $\mathsf{good}_{t-1}$ such that at least half of $j \in [m]$ satisfies $\|B_j^{(t)} - B_i^{(t)}\| \le 8\sqrt{tC}/m$.

**Claim B.2.** *Let $C = \log(16mT/p)$. Then, with probability at least $1 - p/4$, we have*

*(a) for all $i \in \mathsf{good}$ and $t \in [T]$, $\|B_i^{(t)} - B_\star^{(t)}\| \le 4\sqrt{tC}/m$.*

*(b) for all $t \in [T]$, each $i \in \mathsf{good}$ is a valid choice for $B_{\mathsf{med}}^{(t)} = B_i^{(t)}$.*

*(c) for all $i \in \mathsf{good}$ and $t \in [T]$, $\|B_i^{(t)} - B_{\mathsf{med}}^{(t)}\| \le 16\sqrt{tC}/m$ and $\|B_\star^{(t)} - B_{\mathsf{med}}^{(t)}\| \le 12\sqrt{tC}/m$*

*(d) for all $i \in \mathsf{good}$ and $t \in [T]$, we also have $i \in \mathsf{good}_{t+1}$.*

*(e) $\big\|\sum_{i\in\mathsf{good}} \big(B_i^{(t)} - B_\star^{(t)}\big)\big\| \le O(\sqrt{t\log(T/p)}/\sqrt{m})$.*

*Proof of Claim B.2.* We prove by induction. Suppose the statements hold for $t-1$ and we now move to $t$.

(a) For each $i \in \mathsf{good}$, note $\mathbb{E}[\nabla_{t,i}] = \nabla_t$ and $\|\nabla_{t,i} - \nabla_t\| \le 1$. Let $X_t = \frac{\nabla_{t,i} - \nabla_t}{|\mathsf{good}_t|}$, so that $\|X_t\| \le \frac{1}{|\mathsf{good}_t|} \le \frac{1}{(1-\alpha)m} \le \frac{2}{m}$. We can thus apply Lemma B.1 to the $X_t$ and then take a union bound over all $i \in \mathsf{good}$. Thus, with probability at least $1 - \frac{p}{8T}$ we have $\|B_i^{(t)} - B_\star^{(t)}\| \le 4\sqrt{tC}/m$ for all $i \in \mathsf{good}$. The result follows from a further union bound over $t \in [T]$.

(b) Claim B.2a implies for every $i, j \in \mathsf{good}$ we have $\|B_i^{(t)} - B_j^{(t)}\| \le 8\sqrt{tC}/m$. Therefore each $i \in \mathsf{good}$ is a valid choice for setting $B_{\mathsf{med}}^{(t)} = B_i^{(t)}$.

(c) This is a consequence of the previous items and the definition of $B_{\mathsf{med}}^{(t)}$.

(d) This is a consequence of the previous item.

(e) We can apply Lemma B.1 with $\{X_1, X_2, \dots, X_{t|\mathsf{good}|}\} = \{\frac{\nabla_{k,i} - \nabla f(w_k)}{|\mathsf{good}_k|}\}_{k\in[t], i\in\mathsf{good}}$. It holds with probability at least $1 - \frac{p}{8T}$ that $\big\|\sum_{i\in\mathsf{good}} \big(B_i^{(t)} - B_\star^{(t)}\big)\big\| \le O(\sqrt{t\log(T/p)}/\sqrt{m})$.

□

*Proof of Lemma 4.2.* The property $\mathsf{good}_t \supseteq \mathsf{good}$ is from Claim B.2d.

The property $\|\sigma_t\|^2 \le O(\frac{\log(T/p)}{m})$ is by standard concentration inequalities for sums of bounded random vectors.

The property $\|\sigma_0 + \cdots + \sigma_{t-1}\|^2 \le O(\frac{T\log(T/p)}{m})$ is from Claim B.2e.

The property $\|\Delta_t\| \le \alpha$ is obvious as we have at most $\alpha$ fraction of the bad machines.

The bound on $\|\Delta_0 + \cdots + \Delta_{t-1}\|^2$ can be derived as follows. For every $i \in [m] \setminus \mathsf{good}$, let $t$ be the last iteration $i$ satisfies $i \in \mathsf{good}_t$. Then, by the triangle inequality,

$$\|B_i^{(t+1)} - B_\star^{(t+1)}\| \le \frac{2}{m} + \|B_i^{(t)} - B_\star^{(t)}\|$$

On the other hand, $t \in \mathsf{good}_t$ implies $\|B_i^{(t)} - B_{\mathsf{med}}^{(t)}\| \le 16\sqrt{tC}/m$ by the algorithm; combining this with $\|B_\star^{(t)} - B_{\mathsf{med}}^{(t)}\| \le 12\sqrt{tC}/m$, and summing up over all such bad machines $i$ finishes the proof.

The final two properties follow from standard facts about Gaussian random vectors. □

## B.2 Proof of Lemma 4.3

*Proof of Lemma 4.3.* Using the Lipschitz smoothness of $f(\cdot)$, we have

$$\begin{aligned} f(w_t) - f(w_{t+1}) &\ge \langle \nabla f(w_t), w_t - w_{t+1}\rangle - \frac{1}{2}\|w_t - w_{t+1}\|^2 \\ &= \eta\|\nabla f(w_t)\|^2 + \eta\langle \nabla f(w_t), \Xi_t\rangle - \frac{1}{2}\|w_t - w_{t+1}\|^2 + \eta\langle \nabla f(w_t), \xi_t\rangle \end{aligned}$$

We first show:

$$\|w_t - w_{t+1}\|^2 = \eta^2\|\nabla f(w_t) + \Xi_t - \xi_t\|^2 \le 3\eta^2(\|\nabla f(w_t)\|^2 + \|\Xi_t\|^2 + \|\xi_t\|^2)$$

$$|\sum_{t=0}^{T-1} \eta\langle \nabla f(w_t), \xi_t\rangle| \le \eta\sqrt{\sum_{t=0}^{T-1} \|\nabla f(w_t)\|^2} \cdot O(\nu\sqrt{C_1}) \le \left(0.05\eta \sum_{t=0}^{T-1} \|\nabla f(w_t)\|^2\right) + O(C_1\nu^2\eta)$$

The first follows since $(a+b+c)^2 \le 3(a^2+b^2+c^2)$ for any $a, b, c \in \mathsf{R}$, and the second follows from Lemma 4.2. Combining them, and also using that $\|\Xi_t\|^2 \le O(C_2)$, $\|\xi_t\|^2 \le O(d\nu^2 C_1)$, and $\eta \le 0.01$, we

have

$$f(w_0) - f(w_T) \geq 0.9\eta \sum_{t=0}^{T-1} \left( \|\nabla f(w_t)\|^2 - O(\eta C_2) \right) + \eta \sum_{t=0}^{T-1} \langle \nabla f(w_t), \Xi_t \rangle - O(\eta T \nu^2 C_1 (\eta d + \frac{1}{T})) \tag{B.1}$$

For the inner product on the right hand of (B.1), we have that

$$\eta \sum_{t=0}^{T-1} \langle \nabla f(w_t), \Xi_t \rangle = \underbrace{\frac{\eta}{T} \sum_{q=0}^{T-1} \Big\langle \nabla f(w_q), \sum_{t=0}^{T-1} \Xi_t \Big\rangle}_{\spadesuit} + \underbrace{\frac{\eta}{T} \sum_{q=0}^{T-1} \sum_{t=0}^{T-1} \langle \nabla f(w_t) - \nabla f(w_q), \Xi_t \rangle}_{\clubsuit} \tag{B.2}$$

For the first term $\spadesuit$, we have

$$\begin{aligned}
|\spadesuit| \leq \frac{\eta}{T} \sum_{q=0}^{T-1} \Big| \Big\langle \nabla f(w_q), \sum_{t=0}^{T-1} \Xi_t \Big\rangle \Big| &\leq \frac{\eta}{T} \sum_{q=0}^{T-1} \|\nabla f(w_q)\| \cdot \Big\| \sum_{t=0}^{T-1} \Xi_t \Big\| \\
&\leq 0.1\eta \sum_{q=0}^{T-1} \|\nabla f(w_q)\|^2 + \frac{O(\eta)}{T^2} \sum_{q=0}^{T-1} \Big\| \sum_{t=0}^{T-1} \Xi_t \Big\|^2 \\
&\leq 0.1\eta \sum_{q=0}^{T-1} \|\nabla f(w_q)\|^2 + O\Big(\eta C_2\Big)
\end{aligned}$$

where the last inequality follows from Lemma 4.2.

For the second term $\clubsuit$, we have

$$\begin{aligned}
|\clubsuit| \leq \frac{\eta}{T} \sum_{q=0}^{T-1} \Big| \sum_{t=0}^{T-1} \langle \nabla f(w_t) - \nabla f(w_q), \Xi_t \rangle \Big| &\leq \underbrace{\frac{\eta}{T} \sum_{q=0}^{T-1} \Big| \sum_{t=0}^{T-1} \langle \nabla^2 f(w_0)(w_t - w_q), \Xi_t \rangle \Big|}_{\diamondsuit} \\
&+ \underbrace{\frac{\eta}{T} \sum_{q=0}^{T-1} \sum_{t=0}^{T-1} (\|w_t - w_0\| + \|w_q - w_0\|) \|w_t - w_q\| \|\Xi_t\|}_{\heartsuit}
\end{aligned}$$

Using $\|w_t - w_q\| \leq \|w_t - w_0\| + \|w_q - w_0\|$, one can derive

$$\begin{aligned}
\heartsuit &\leq \frac{\eta}{T} \sum_{q=0}^{T-1} \sum_{t=0}^{T-1} (\|w_t - w_0\| + \|w_q - w_0\|)^2 \cdot O(\sqrt{C_2}) \\
&\leq \eta \sum_{t=0}^{T-1} \|w_t - w_0\|^2 \cdot O(\sqrt{C_2}) \\
&\leq \eta^3 \sum_{t=0}^{T-1} \|\nabla f(w_0) + \cdots + \nabla f(w_{t-1}) + \Xi_0 + \cdots + \Xi_{t-1} + \xi_0 + \cdots + \xi_{t-1}\|^2 \cdot O(\sqrt{C_2}) \\
&\leq O(\sqrt{C_2}\eta^3 T^2) \sum_{t=0}^{T-1} \|\nabla f(w_t)\|^2 + O(\sqrt{C_2} C_2 \eta^3 T^2) + O(\eta^3 \nu^2 T^2 d C_1 \sqrt{C_2})
\end{aligned}$$

As for $\diamondsuit$,

$$\Big| \sum_{t=0}^{T-1} \langle \nabla^2 f(w_0)(w_t - w_q), \Xi_t \rangle \Big| \leq \Big| \sum_{t=q+1}^{T-1} \langle \nabla^2 f(w_0)(w_t - w_q), \Xi_t \rangle \Big| + \Big| \sum_{t=0}^{q-1} \langle \nabla^2 f(w_0)(w_t - w_q), \Xi_t \rangle \Big|$$

For the first term (and the second term is analogous), we have

$$\begin{aligned}
&\Big| \sum_{t=q+1}^{T-1} \langle \nabla^2 f(w_0)(w_t - w_q), \Xi_t \rangle \Big| \\
&= \eta \Big| \sum_{t=q+1}^{T-1} \langle \nabla^2 f(w_0)(\nabla f(w_q) + \cdots \nabla f(w_{t-1}) + \Xi_q + \cdots \Xi_{t-1} + \xi_q + \cdots + \xi_{t-1}), \Xi_t \rangle \Big| \\
&\leq \eta \Big| \sum_{t=q+1}^{T-1} \langle \nabla^2 f(w_0)(\xi_q + \cdots + \xi_{t-1}), \Xi_t \rangle \Big| +
\end{aligned}$$

$$\eta\Big|\sum_{t=q+1}^{T-1}\langle\nabla^2 f(w_0)(\nabla f(w_q)+\cdots\nabla f(w_{t-1})),\Xi_t\rangle\Big|+\eta\Big|\sum_{t=q+1}^{T-1}\langle\nabla^2 f(w_0)(\Xi_q+\cdots\Xi_{t-1}),\Xi_t\rangle\Big|$$

$$\leq \eta\cdot O(\sqrt{d\nu^2 TC_1}\cdot\sqrt{TC_2})+ \qquad \text{(using Lemma 3.2)}$$

$$\eta\Big|\sum_{t=q}^{T-2}\langle\nabla^2 f(w_0)\nabla f(w_t)\,,\Xi_{t+1}+\cdots+\Xi_{T-1}\rangle\Big|+\frac{\eta}{2}\Big\langle\nabla^2 f(w_0)(\Xi_q+\cdots\Xi_{T-1}),(\Xi_q+\cdots\Xi_{T-1})\Big\rangle$$

$$\leq \eta\cdot O(\sqrt{d\nu^2 TC_1}\cdot\sqrt{TC_2})+\eta\sum_{t=q}^{T-2}\|\nabla f(w_t)\|\|\Xi_{t+1}+\cdots+\Xi_{T-1}\|+\frac{\eta}{2}\|\Xi_q+\cdots\Xi_{T-1}\|^2$$

$$\leq O(\eta\sqrt{TC_2})\cdot\sum_{t=0}^{T-1}\|\nabla f(w_t)\|+O(T\eta C_2+T\eta\nu^2 dC_1)\ ,$$

where the last inequality follows from Young's inequality, and Lemma 4.2. Putting this back to the formula of $\diamondsuit$, we have

$$\begin{aligned}\diamondsuit &\leq O(\eta^2\sqrt{TC_2})\cdot\sum_{t=0}^{T-1}\|\nabla f(w_t)\|+O(T\eta^2 C_2+T\eta^2\nu^2 dC_1)\\ &\leq 0.1\eta\sum_{t=0}^{T-1}\|\nabla f(w_t)\|^2+O(\eta^3T^2C_2+T\eta^2C_2+T\eta^2\nu^2dC_1)\end{aligned}$$

Finally, putting $\diamondsuit$ and $\heartsuit$ back to $\clubsuit$, and putting $\clubsuit$ and $\spadesuit$ back to (B.2) and (B.1), we have

$$\begin{aligned}f(w_0)-f(w_T)\geq{}& 0.8\eta\sum_{t=0}^{T-1}\|\nabla f(w_t)\|^2-O(\sqrt{C_2}\eta^3T^2)\sum_{t=0}^{T-1}\|\nabla f(w_t)\|^2\\ &-C_2\cdot O(\eta+\eta^2T+\eta^3T^2+\sqrt{C_2}\eta^3T^2)-C_1\cdot O(T\eta^2\nu^2d+T^2\eta^3\nu^2\sqrt{C_2}+\eta T\nu^2(\eta d+\frac{1}{T}))\end{aligned}$$

together with $T=\frac{1}{100\eta(1+\sqrt{C_2})}$ and $\eta\leq 0.01\min\{1,\frac{1}{C_2}\}$, we have

$$\begin{aligned}f(w_0)-f(w_T)&\geq 0.7\eta\sum_{t=0}^{T-1}\|\nabla f(w_t)\|^2-C_2\cdot O(\eta+\eta^2T+\eta^3T^2+\sqrt{C_2}\eta^3T^2)-C_1\cdot O(T\eta\nu^2\eta(d+\sqrt{C_2}))\\ &=0.7\eta\sum_{t=0}^{T-1}\Big(\|\nabla f(w_t)\|^2-C_2\cdot O(\frac{1}{T}+\eta+\eta^2T+\sqrt{C_2}\eta^2T)-C_1\cdot O(\eta T\nu^2\eta(d+\sqrt{C_2}))\Big)\\ &\geq 0.7\eta\sum_{t=0}^{T-1}\Big(\|\nabla f(w_t)\|^2-\eta\cdot O(C_2+(C_2)^{1.5})-O(C_1\nu^2\eta(d+\sqrt{C_2}))\Big)\ .\end{aligned}$$

□

### B.3 Proof of Lemma 4.4

*Proof of Lemma 4.4.* Let us denote by $r_t=\frac{[\xi_t^{\mathsf{a}}]_1}{2}=-\frac{[\xi_t^{\mathsf{b}}]_1}{2}$ and we know $r_t\sim\mathcal{N}(0,\frac{\nu^2}{4})$. We can write

$$w_{t+1}^{\mathsf{a}}-w_{t+1}^{\mathsf{b}}=\eta r_t\mathbf{e}_1+w_t^{\mathsf{a}}-w_t^{\mathsf{b}}-\eta(\nabla f(w_t^{\mathsf{a}})-\nabla f(w_t^{\mathsf{b}}))-\eta(\Xi_t^{\mathsf{a}}-\Xi_t^{\mathsf{b}})$$

Using the second-order smoothness, we have

$$\begin{aligned}\nabla f(w_t^{\mathsf{a}})-\nabla f(w_t^{\mathsf{b}})&=\int_{\tau=0}^1\nabla^2 f\big(w_t^{\mathsf{a}}+\tau(w_t^{\mathsf{b}}-w_t^{\mathsf{a}})\big)(w_t^{\mathsf{a}}-w_t^{\mathsf{b}})d\tau\\ &=\nabla^2 f(w_0)\cdot(w_t^{\mathsf{a}}-w_t^{\mathsf{b}})+\theta_t\end{aligned}$$

for some vector $\|\theta_t\|\leq\max\{\|w_0^{\mathsf{a}}-w_t^{\mathsf{a}}\|,\|w_0^{\mathsf{b}}-w_t^{\mathsf{b}}\|\}\cdot\|w_t^{\mathsf{a}}-w_t^{\mathsf{b}}\|$. Therefore, we have

$$w_{t+1}^{\mathsf{a}}-w_{t+1}^{\mathsf{b}}=\eta r_t\mathbf{e}_1+\big(\mathbf{I}-\eta\nabla^2 f(w_0)\big)(w_t^{\mathsf{a}}-w_t^{\mathsf{b}})-\eta(\Xi_t^{\mathsf{a}}-\Xi_t^{\mathsf{b}}+\theta_t)$$

Now, giving $\psi_0=\overline{\psi}_0=0$, imagine two sequences

- $\psi_{t+1}=\eta r_t\mathbf{e}_1+\big(\mathbf{I}-\eta\nabla^2 f(w_0)\big)\psi_t$ and
- $\overline{\psi}_{t+1}=\eta r_t\mathbf{e}_1+\big(\mathbf{I}-\eta\nabla^2 f(w_0)\big)\overline{\psi}_t-\eta(\Xi_t^{\mathsf{a}}-\Xi_t^{\mathsf{b}}+\theta_t)=w_{t+1}^{\mathsf{a}}-w_{t+1}^{\mathsf{b}}$

We will inductively prove $\|\psi_t-\overline{\psi}_t\|\leq\frac{1}{2}\|\psi_t\|$. On one hand, it is easy to see that $\psi_t$ is zero except in the first coordinate, in which it behaves as a Gaussian with zero mean and variance $\sum_{k=0}^{t-1}(1+\eta\delta)^{2k}\cdot\frac{\eta^2\nu^2}{4}=$

$\Theta\left(\frac{(1+\eta\delta)^{2t}}{\eta\delta}\cdot\eta^2\nu^2\right)$. By Gaussian tail bounds, we know that

- with probability at least 0.99, it satisfies $\|\psi_t\| \le O(\frac{\sqrt{\eta C_1}\nu(1+\eta\delta)^t}{\sqrt{\delta}})$ for every $t$
- with probability at least 0.99, it satisfies $\|\psi_T\| \ge \frac{1}{1000}(\frac{\sqrt{\eta}\nu(1+\eta\delta)^T}{\sqrt{\delta}})$

In the rest of the proof, we condition on this event happens. We prove towards contradiction by assuming $\|w_t^{\mathsf{a}} - w_0^{\mathsf{a}}\| \le R$ and $\|w_t^{\mathsf{b}} - w_0^{\mathsf{b}}\| \le R$ for all $t \in [T]$.

We will inductively prove that $\|\psi_t - \overline{\psi}_t\| \le \frac{1}{2000}(\frac{\sqrt{\eta}\nu(1+\eta\delta)^t}{\sqrt{\delta}})$. We calculate the difference

$$\psi_t - \overline{\psi}_t = \eta\sum_{i=0}^{t-1}\left(\mathbf{I} - \eta\nabla^2 f(w_0)\right)^{t-1-i}(\Xi_i^{\mathsf{a}} - \Xi_i^{\mathsf{b}} + \theta_i)$$

Let $g = \frac{\psi_t - \overline{\psi}_t}{\|\psi_t - \overline{\psi}_t\|}$, then we can inner-product the above equation by vector $g$, which gives

$$\begin{aligned}
\|\psi_t - \overline{\psi}_t\| &= \eta\sum_{i=0}^{t-1}\left\langle \Xi_i^{\mathsf{a}} - \Xi_i^{\mathsf{b}} + \theta_i\,,\left(\mathbf{I} - \eta\nabla^2 f(w_0)\right)^{t-1-i} g\right\rangle \\
&\overset{①}{\le} \eta\sum_{i=0}^{t-1}\left(\left\langle \Xi_i^{\mathsf{a}} - \Xi_i^{\mathsf{b}}\,,\left(\mathbf{I} - \eta\nabla^2 f(w_0)\right)^{t-1-i} g\right\rangle + R\cdot O(\frac{\sqrt{\eta C_1}\nu(1+\eta\delta)^i}{\sqrt{\delta}})\cdot(1+\eta\delta)^{t-1-i}\right) \\
&\le \eta\sum_{i=0}^{t-1}\left\langle \Xi_i^{\mathsf{a}} - \Xi_i^{\mathsf{b}}\,,\left(\mathbf{I} - \eta\nabla^2 f(w_0)\right)^{t-1-i} g\right\rangle + O(R\eta T\frac{\sqrt{\eta C_1}}{\sqrt{\delta}}\nu(1+\eta\delta)^t)
\end{aligned}$$

where the inequality ① uses $\|\theta_i\| \le R\cdot\|\overline{\psi}_i\| \le R\cdot(\|\psi_i\| + \|\psi_i - \overline{\psi}_i\|)$, $\left\|\left(\mathbf{I} - \eta\nabla^2 f(w_0)\right)^{t-1-i} g\right\| \le (1+\eta\delta)^{t-1-i}$, and the inductive assumption. Let us call $M = \left(\mathbf{I} - \eta\nabla^2 f(w_0)\right)$, and focus on

$$\begin{aligned}
&\left|\sum_{i=0}^{t-1}\left\langle \Xi_i^{\mathsf{a}}\,,\left(\mathbf{I} - \eta\nabla^2 f(w_0)\right)^{t-1-i} g\right\rangle\right| \\
&= \left|\langle \Xi_0^{\mathsf{a}} + \cdots + \Xi_{t-1}^{\mathsf{a}}\,, g\rangle + \sum_{i=0}^{t-2}\left\langle \Xi_0^{\mathsf{a}} + \cdots + \Xi_i^{\mathsf{a}}\,, M^{t-1-i}g - M^{t-2-i}g\right\rangle\right| \\
&\le \|\Xi_0^{\mathsf{a}} + \cdots + \Xi_{t-1}^{\mathsf{a}}\|\cdot\|g\| + \sum_{i=0}^{t-2}\|\Xi_0^{\mathsf{a}} + \cdots + \Xi_i^{\mathsf{a}}\|\cdot\|M^{t-1-i}g - M^{t-2-i}g\| \\
&\le O(\sqrt{TC_2}\left(\|g\| + \sum_{i=0}^{t-2}\|M^{t-1-i}g - M^{t-2-i}g\|\right) \qquad \text{(using Lemma 4.2)} \\
&\le O(\sqrt{TC_2}\cdot(1+\eta\delta)^{t-1})
\end{aligned}$$

Together, we have

$$\|\psi_t - \overline{\psi}_t\| \le O(\eta\sqrt{TC_2})\cdot(1+\eta\delta)^t + O(R\eta T\frac{\sqrt{\eta C_1}}{\sqrt{\delta}}\nu(1+\eta\delta)^t)$$

Under our assumption, we have $\|\psi_t - \overline{\psi}_t\| < \frac{1}{2000}(\frac{\sqrt{\eta}\nu(1+\eta\delta)^t}{\sqrt{\delta}})$ and therefore $\|\overline{\psi}_T\| \ge \|\psi_T\| - \|\psi_T - \overline{\psi}_T\| \ge \frac{1}{2000}(\frac{\sqrt{\eta}\nu(1+\eta\delta)^t}{\sqrt{\delta}})$. Thus, within $T$ iterations, we have $\|\overline{\psi}_t\| > R$ and this gives a contradiction. □

# C FINAL: DOUBLE SAFE GUARD

We now come to our final Algorithm 3 which is our perturbed SGD algorithm with two safeguards. The two safeguard algorithm naturally divides itself into epochs, each consisting of $T_1$ iterations. We will demonstrate that within most epochs, we make good progress. Thus, consider some iterate $x_{mT_1}$, for some $m < T/T_1$. Our goal will be to argue that we make good function value progress by iterate $x_{(m+1)T_1}$, and that we do not settle into any saddle points. To slightly simplify notation, let $w_0 = x_{mT_1}$, and let the sequence of iterates be $w_0, \ldots, w_{T_1-1}$, so that $w_{T_1-1} = x_{(m+1)T_1-1}$. For completeness' sake we rewrite this as Algorithm 1.

Our main result is the following theorem.

**Theorem C.1.** *Let $C_3 = \alpha^2 + \frac{1}{m}$. Suppose we pick parameters $p, \delta \in (0,1)$, $\eta \le \widetilde{O}(\frac{\delta^3}{C_3})$, $\nu^2 = \widetilde{\Theta}(C_3)$, $T_0 = \widetilde{\Theta}\big(\frac{1}{\eta}\big)$, $T_1 = \widetilde{\Theta}\big(\frac{1}{\eta\delta}\big) \ge T_0$, $\mathfrak{T}_1 = \widetilde{\Theta}(\sqrt{T_1})$, and $\mathfrak{T}_0 = \widetilde{\Theta}(\sqrt{T_0})$. Then, starting from $w_0$,*

**Algorithm 3** Perturbed SGD with double safe guard (for analysis purpose)

**Input:** $w_0 \in \mathbb{R}^d$, set $\mathsf{good}_0 \supseteq \mathsf{good}$, rate $\eta > 0$, lengths $T_1 \geq T_0 \geq 1$, threshold $\mathfrak{T}_1 > \mathfrak{T}_0 > 0$;

**for** $t \leftarrow 0$ **to** $T_1 - 1$ **do**
  $last \leftarrow \max\{t_0 \in [t] \colon t_0 \text{ is a multiple of } T_0\}$
  **for each** $i \in \mathsf{good}_t$ **do**
    receive $\nabla_{t,i} \in \mathbb{R}^d$ from machine $i$;
    $A_i \leftarrow \sum_{k=0}^{t} \frac{\nabla_{k,i}}{|\mathsf{good}_k|}$ and $B_i \leftarrow \sum_{k=last}^{t} \frac{\nabla_{k,i}}{|\mathsf{good}_k|}$;
  $A_{\mathsf{med}} \leftarrow A_i$ where $i \in \mathsf{good}_t$ is any machine s.t. $\left|\{j \in \mathsf{good}_t \colon \|A_j - A_i\| \leq \mathfrak{T}_1\}\right| > m/2$.
  $B_{\mathsf{med}} \leftarrow B_i$ where $i \in \mathsf{good}_t$ is any machine s.t. $\left|\{j \in \mathsf{good}_t \colon \|B_j - B_i\| \leq \mathfrak{T}_0\}\right| > m/2$.
  $\mathsf{good}_{t+1} \leftarrow \left\{i \in \mathsf{good}_t \colon \|A_i - A_{\mathsf{med}}\| \leq 2\mathfrak{T}_1 \bigwedge \|B_i - B_{\mathsf{med}}\| \leq 2\mathfrak{T}_0\right\}$;
  $w_{t+1} = w_t - \eta\left(\xi_t + \frac{1}{|\mathsf{good}_t|}\sum_{i \in \mathsf{good}_t} \nabla_{t,i}\right)$;

*(a) with probability at least $1-p$ we have*

$$f(w_0) - f(w_{T_1}) \geq 0.7\eta \sum_{t=0}^{T_1-1} \left(\|\nabla f(w_t)\|^2 - \widetilde{O}(\eta C_3 d)\right) \quad .$$

*(b) As long as $\|w_t - w_0\| \geq R$ for some $t \in \{1, 2, \ldots, T_1\}$ and for $R = \widetilde{\Theta}(\delta) \leq \frac{\delta}{2}$, then with probability at least $1-p$ we have then*

$$f(w_0) - f(w_{T_1}) \geq 0.5\eta \sum_{t=0}^{T_1-1} \left(-\widetilde{O}(\eta C_3 d)\right) + \widetilde{\Omega}(\delta^3)$$

*(c) if $\lambda_{\min}(\nabla^2 f(w_0)) \leq -\delta$, we also have with probability at least 0.45,*

$$f(w_0) - f(w_{T_1}) \geq 0.5\eta \sum_{t=0}^{T_1-1} \left(-\widetilde{O}(\eta C_3 d)\right) + \widetilde{\Omega}(\delta^3)$$

### C.1 Why Theorem C.1 Implies Theorem 2.3

Using the parameter choice $\eta = \widetilde{\Theta}(\frac{\varepsilon^2}{C_3 d})$ from Theorem 2.3, we know $\widetilde{O}(\eta C_3 d) \leq 0.1\varepsilon^2$. We claim two things:

- For at least $90\%$ of the epochs, they must satisfy (denoting by $w_0$ and $w_{T_1}$ the beginning and ending points of this epoch)

  $$f(w_0) - f(w_{T_1}) \leq 20\frac{f(x_0) - \min f(x)}{T/T_1} \leq \varepsilon^{1.5}$$

  The last inequality uses our choice of $T$ and $\delta = \widetilde{\Theta}(\sqrt{\varepsilon})$.

  The reason for this is by way of contradiction. Suppose for at least $10\%$ of the epochs it satisfies $f(w_0) - f(w_{T_1}) > 20\frac{f(x_0)-\min f(x)}{T/T_1}$, then, for the remainder of the epochs, they must at least satisfy $f(w_0) - f(w_{T_1}) \geq -0.7\eta T_1 \cdot 0.1\varepsilon^2$. Summing over all the epochs, we shall obtain $f(x_0) - f(x_T) > f(x_0) - \min f(x)$ but this gives a contradiction.
- For at least $40\%$ of the epochs, they must satisfy the three properties from Theorem C.1.

In particular, for at least $30\%$ of the epochs, they must satisfy both. Since $\varepsilon^{1.5}$ is so small that

$$\varepsilon^{1.5} \geq f(w_0) - f(w_{T_1}) \geq 0.5\eta \sum_{t=0}^{T_1-1} \left(-\widetilde{O}(\eta C_3 d)\right) + \widetilde{\Omega}(\delta^3) \geq \widetilde{\Omega}(\delta^3) - 0.05\eta T_1 \varepsilon^2$$

would give a contradiction (for instance, one can choose $\delta$ to be slightly larger than $\sqrt{\varepsilon}$ by some log factors), this means, for those $30\%$ of the epochs, they must satisfy:

- $\varepsilon^{1.5} \geq 0.7\eta \sum_{t=0}^{T_1-1} \left(\|\nabla f(w_t)\|^2 - 0.1\varepsilon^2\right)$ ,
- $\|w_t - w_0\| \leq \frac{\delta}{2}$ for every $t = 1, 2, \ldots, T_1$, and
- $\nabla^2 f(w_0) \succeq -\delta \mathbf{I}$.

The latter two properties together implies $\nabla^2 f(w_t) \succeq -\frac{\delta}{2}\mathbf{I}$ for every $t = 1, 2, \ldots, T_1$ (by the second-order smoothness). The first property implies for at least $90\%$ of the iterations $t$ in this epoch, they must satisfy $\|\nabla f(x)\| \leq \varepsilon$. This finishes the proof of Theorem 2.3. ■

### C.2 PROOF OF THEOREM C.1

We first have the following lemma

**Lemma C.2** (double safe guard). *In Algorithm 3, suppose* $\mathfrak{T}_1 = 8\sqrt{T_1 \log(16mT_1/p)}$ *and* $\mathfrak{T}_0 = 8\sqrt{T_0 \log(16mT_1/p)}$. *Then, with probability at least* $1 - p/2$, *for every* $t = 0, \ldots, T_1 - 1$,

- $\mathsf{good}_t \supseteq \mathsf{good}$.
- $\|\sigma_t\|^2 \le O(\frac{\log(T_1/p)}{m})$, $\|\Delta_t\|^2 \le \alpha^2$, $\|\xi_t\|^2 \le O(\nu^2 d \log(T_1/p))$,
- $\|\sigma_0 + \cdots + \sigma_{t-1}\|^2 \le O(\frac{T_1 \log(T_1/p)}{m})$, $\|\sigma_{last} + \cdots + \sigma_{t-1}\|^2 \le O(\frac{T_0 \log(T_1/p)}{m})$
- $\|\Delta_0 + \cdots + \Delta_{t-1}\|^2 \le O(\alpha^2 T_1 \log(mT_1/p))$ *and* $\|\Delta_{last} + \cdots + \Delta_{t-1}\|^2 \le O(\alpha^2 T_0 \log(mT_1/p))$
- $\|\xi_0 + \cdots + \xi_{t-1}\|^2 \le O(\nu^2 d T_1 \log(T_1/p))$ *and* $\|\xi_{last} + \cdots + \xi_{t-1}\|^2 \le O(\nu^2 d T_0 \log(T_1/p))$.

*We call this probabilistic event* $\mathsf{Event}^{\mathsf{double}}_{T_1,T_0}(w_0)$ *and* $\Pr[\mathsf{Event}^{\mathsf{double}}_{T_1,T_0}(w_0)] \ge 1 - p/2$.

The proof is a direct corollary of Lemma 4.2, by combining events $\mathsf{Event}^{\mathsf{single}}_{T_1}(w_0)$, $\mathsf{Event}^{\mathsf{single}}_{T_0}(w_0)$, $\mathsf{Event}^{\mathsf{single}}_{T_0}(w_{T_0})$, $\mathsf{Event}^{\mathsf{single}}_{T_0}(w_{2T_0})$ and so on. The next lemma is a simple corollary by repeatedly applying Lemma 4.3. It proves Theorem C.1a.

**Lemma C.3** (modified from Lemma 4.3). *Denote by* $C_1 = \log(T_1/p)$ *and* $C_2 = \alpha^2 \log \frac{mT_1}{p} + \frac{\log(T_1/p)}{m}$. *Suppose* $\eta \le 0.01 \min\{1, \frac{1}{C_2}\}$, $T_0 = \frac{1}{100\eta(1+\sqrt{C_2})}$ *and* $T_1 \ge T_0$. *We start from* $w_0$ *and apply Algorithm 3. Under event* $\mathsf{Event}^{\mathsf{double}}_{T_1,T_0}(w_0)$, *it satisfies*

$$f(w_0) - f(w_{T_1}) \ge 0.7\eta \sum_{t=0}^{T_1-1} \Big( \|\nabla f(w_t)\|^2 - \eta \cdot O(C_2 + (C_2)^{1.5}) - O(C_1 \nu^2 \eta (d + \sqrt{C_2})) \Big)$$

The next lemma can be easily derived from Lemma 4.4.

**Lemma C.4** (modified from Lemma 4.4). *Suppose*

$$R = \Theta(\frac{\delta}{\sqrt{C_1} \log(\delta^3/\eta C_2)}) \quad \textit{and} \quad \nu^2 = \Theta(C_2 \log \frac{\delta^3}{\eta C_2})$$

*Suppose* $\eta \le 0.01 \min\{1, \frac{\delta^3}{C_2}\}$, $T_0 = \frac{1}{100\eta(1+\sqrt{C_2})}$ *and* $T_1 = \Theta(\frac{1}{\eta\delta} \log \frac{\delta^3}{\eta C_2}) \ge T_0$. *Let* $w_0 \in \mathbb{R}^d$ *be any point in the space and suppose* $\lambda_{\min}(\nabla^2 f(w_0)) \le -\delta$ *for some* $\delta \ge 0$. *Given two coupled sequences defined as before, under events* $\mathsf{Event}^{\mathsf{double}}_{T_1,T_0}(w_0^{\mathsf{a}})$ *and* $\mathsf{Event}^{\mathsf{double}}_{T_1,T_0}(w_0^{\mathsf{b}})$, *we have with probability at least 0.98*

$$\begin{aligned} &\max\left\{ f(w_0^{\mathsf{a}}) - f(w_{T_1}^{\mathsf{a}}), f(w_0^{\mathsf{b}}) - f(w_{T_1}^{\mathsf{b}}) \right\} \\ \ge\ & 0.5\eta \sum_{t=0}^{T_1-1} \Big( -\eta \cdot O(C_2 + (C_2)^{1.5}) - O(C_1 \nu^2 \eta (d + \sqrt{C_2})) \Big) + \Omega(\frac{\delta^3}{C_1 \log^3 \frac{\delta^3}{\eta C_2}}) \end{aligned}$$

Lemma C.4 directly proves the second half of Theorem C.1c, because given two coupled sequences with the same marginal distribution, we have

$$\Pr[f(w_0^{\mathsf{a}}) - f(w_{T_1}^{\mathsf{a}}) \ge X] \ge \frac{1}{2} \Pr[\max\left\{ f(w_0^{\mathsf{a}}) - f(w_{T_1}^{\mathsf{a}}), f(w_0^{\mathsf{b}}) - f(w_{T_1}^{\mathsf{b}}) \right\} \ge X]$$

*Proof of Lemma C.4.* Our choice on $r$ and $R$ satisfy the requirement of Lemma 4.4. Suppose without loss of generality that the $w_t^{\mathsf{a}}$ sequence leaves $w_0$ by more than $R$. Let $T_1^{\mathsf{a}}$ be the first iteration $t \le T_1$ in which $\|w_t^{\mathsf{a}} - w_0^{\mathsf{a}}\| \ge R$.

$$\begin{aligned} \|w_{T_1^{\mathsf{a}}}^{\mathsf{a}} - w_0^{\mathsf{a}}\|^2 &= \eta^2 \|\nabla f(w_0^{\mathsf{a}}) + \cdots + \nabla f(w_{T_1^{\mathsf{a}}-1}^{\mathsf{a}}) + \Xi_0^{\mathsf{a}} + \cdots + \Xi_{T_1^{\mathsf{a}}-1}^{\mathsf{a}} + \xi_0^{\mathsf{a}} + \cdots + \xi_{T_1^{\mathsf{a}}-1}^{\mathsf{a}}\|^2 \\ &\le O(\eta^2 T_1) \sum_{t=0}^{T_1^{\mathsf{a}}-1} \|\nabla f(w_t^{\mathsf{a}})\|^2 + O(C_2 \eta^2 T_1) + O(C_1 \eta^2 \nu^2 T_1 d) \end{aligned}$$

Combining this with Lemma C.3, we have

$$\begin{aligned} f(w_0^{\mathsf{a}}) - f(w_{T_1^{\mathsf{a}}}^{\mathsf{a}}) &\ge 0.5\eta \sum_{t=0}^{T_1^{\mathsf{a}}-1} \Big( \|\nabla f(w_t^{\mathsf{a}})\|^2 - \eta \cdot O(C_2 + (C_2)^{1.5}) - O(C_1 \nu^2 \eta (d + \sqrt{C_2})) \Big) + \frac{\|w_{T_1^{\mathsf{a}}}^{\mathsf{a}} - w_0^{\mathsf{a}}\|^2}{100\eta T_1} \\ &\ge 0.5\eta \sum_{t=0}^{T_1^{\mathsf{a}}-1} \Big( \|\nabla f(w_t^{\mathsf{a}})\|^2 - \eta \cdot O(C_2 + (C_2)^{1.5}) - O(C_1 \nu^2 \eta (d + \sqrt{C_2})) \Big) + \frac{R^2}{100\eta T_1} \end{aligned}$$

Combining this with Lemma C.3 again but for the remainder iterations, we have

$$f(w_0^{\mathsf{a}}) - f(w_{T_1}^{\mathsf{a}}) \geq 0.5\eta \sum_{t=0}^{T_1-1} \left( \|\nabla f(w_t^{\mathsf{a}})\|^2 - \eta \cdot O(C_2 + (C_2)^{1.5}) - O(C_1 \nu^2 \eta (d + \sqrt{C_2})) \right) + \frac{R^2}{100\eta T_1}$$

□

In fact, the above same proof of Lemma C.4 also implies Theorem C.1b. These together finish the proof of Theorem C.1. ■

# D More on Experiments

We conduct experiments on training a deep residual network (ResNet-20) He et al. (2016) on the CIFAR-10 image classification task Krizhevsky et al. (2014).

## D.1 Setting and Implemented Methods

In all of our experiments, we use 10 workers and the mini-batch size (per worker) is 8. We run all algorithms for 200 epochs, with initial learning rate $\eta = 0.1$, but the learning rate decreases by a factor of 10 on epochs 80, 120, and 160.

The *ideal baseline* is gradient "mean without attacks", and we compare against *(Naive) Mean*, *Geometric Median* Chen et al. (2017), *Coordinate-wise Median* Yin et al. (2018; 2019), *Krum* Blanchard et al. (2017), and *Zeno* Xie et al. (2018b) with attacks. We set $\alpha = 0.4$ so there are 4 Byzantine workers. (This exceeds the fault-tolerance of Krum, and so we also tested Krum with only 3 Byzantine workers.) We formally define those prior works as follows.

**Definition D.1** (GeoMed Chen et al. (2017))**.** *The geometric median of $\{y_1, ..., y_m\}$, denoted by $geo_med\{y_1, ..., y_m\}$, is*

$$geo_med\{y_1, ..., y_m\} := \arg\min_{y \in \mathbb{R}^d} \sum_{i=1}^{m} \|y - y_i\|$$

*In our experiments, we choose the geometric median from set $\{y_1, ..., y_m\}$.*

**Definition D.2** (coordinate-wise median Yin et al. (2018; 2019))**.** *Coordinate-wise median $g = med\{y_1, ..., y_m\}$ is defined as a vector with its k-th coordinate being $g[k] = \textbf{med}\{y_1[k], ..., y_m[k]\}$ for each $k \in [d]$, where **med** is the usual (one-dimensional) median.*

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

$$KR\{y_1, ..., y_m\} := y_k \quad \textit{where} \quad k = \arg\min_{i \in [m]} \sum_{i \to j} \|y_i - y_j\|^2$$

*and $i \to j$ is the indices of the $m - b - 2$ nearest neighbours of $y_i$ in $\{y_1, ..., y_m\}$ by Euclidean distances.*

Note that Krum requires $2b + 2 < m$. So, we have also repeated the experiments for Krum with 3 Byzantine workers (out of 10 workers) to be more fair.

**Definition D.4** (Zeno Xie et al. (2018b))**.**

$$Zeno_b\{y_1, ..., y_m\} = \frac{1}{m-b} \sum_{i=1}^{m-b} \widetilde{y}(i)$$

*where $\{\widetilde{y}(i) : i \in [m]\}$ are the gradient estimators with the $m - b$ highest "scores", and the so-called stochastic descendant score for any gradient estimator $u$, based on the current parameter $x$, learning rate $\eta$, and a constant weight $\rho > 0$, is defined as:*

$$Score_{\eta,\rho}(u, x) = f_r(x) - f_r(x - \eta u) - \rho \|u\|^2$$

*$f_r(x) - f_r(x - \eta u)$ is the estimated descendant of the loss function and $\rho\|u\|^2$ is the magnitude of the update.*

**Safeguard SGD.** To provide the strongest comparison against prior works, we have made universal choices for our safeguard window sizes (across *all* attackers), and provide an automatic way to select the safeguard thresholds.

Specifically, for our algorithm with a single safeguard we have used a universal window size $T = 3$ epochs, and for our algorithm with double safeguards we have used window sizes $T_0 = 1$ epoch and $T_1 = 6$ epochs.

As for the procedure to tune thresholds, for each attacker, we pre-run the experiment for 20 epochs with some sufficiently large pre-chosen threshold value (e.g. $\mathfrak{T} = 100$ or $\mathfrak{T}_0 = 100$ and $\mathfrak{T}_1 = 100$), and then *automatically* select a good final threshold. Let us describe this process in the case of a single safeguard (and double

safeguard is similar). In each iteration, the master examines $\{B_i | i \in \mathsf{good}_t\}$ to determine some minimal $\mathfrak{T}_{temp}$: it satisfies that at least half of the workers satisfy $\|B_i - B_{med}\| \leq \mathfrak{T}_{temp}$ and there exists at least one worker $j \in \mathsf{good}_t$ such that $\|B_j - B_{med}\| > 2\mathfrak{T}_{temp}$. Then, $\mathfrak{T}_{temp}$ is added to the temporary threshold list (after rounding to the first decimal point). In the end, the automated process will recommend the most frequently repeated items in the temporary threshold list, as the final choice of threshold. We emphasize that this procedure only requires knowing the $\alpha < 0.5$ upper bound on the number/fraction of possibly faulty nodes.

## D.2 Experiment Results by Attacks

### D.2.1 Sign-Flipping Attack

Recall that, in a sign-flipping attack, each Byzantine worker sends the negative gradient to the master. The results for this attack are shown in Figure 2. From the plots, one can see that our single and double safe-guard algorithms *both* outperform prior works, even though some of these algorithms make additional assumptions, and closely track the performance of the ideal baseline (without failures). Under this attacker, single and double safeguard algorithms perform similarly.

(Parameters: $m = 10$, $\alpha = 0.4$. Single safe-guard threshold $\mathfrak{T} = 2$, window size $T = 3$ epochs, variance $\nu^2 = 10^{-8}$; double safe-guard thresholds $\mathfrak{T}_0 = 1, \mathfrak{T}_1 = 2$ and window sizes $T_0 = 1$ epoch, $T_1 = 6$ epochs, variance $\nu^2 = 10^{-8}$. Zeno Batch size $n_r = 4$ and constant weight $\rho = 0.0005$.)

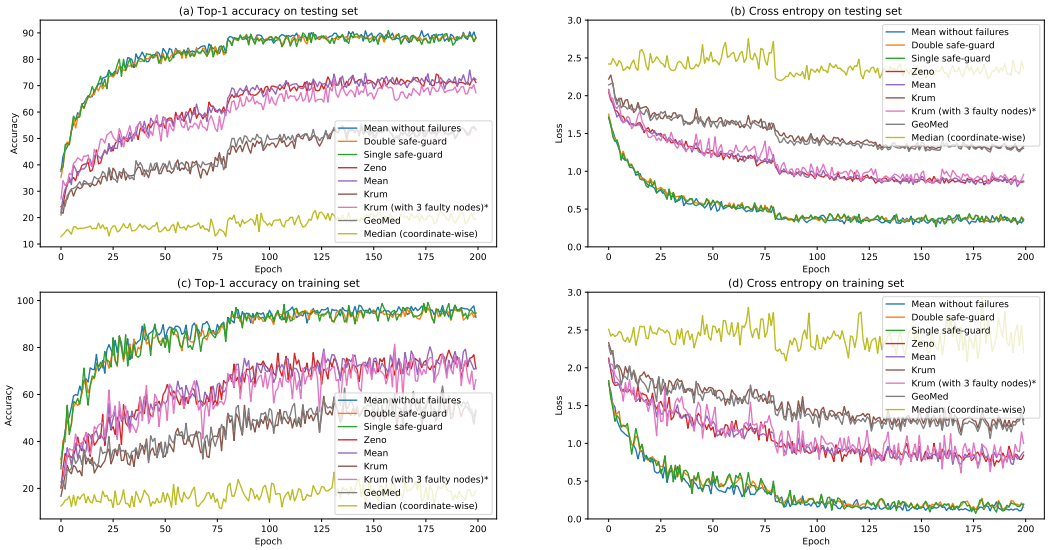


Figure 2: Comparison under sign-flipping attack.

### D.2.2 Delayed-Gradient Attack

Recall that, in a delayed-gradient attack, each Byzantine worker sends an *old* gradient to the master. In our experiments, the delay is of $D = 1000$ iterations. The results are shown in Figure 3. From the plots, one can see that our single and double safe-guard algorithms *both* outperform prior works, and closely track the performance of the ideal baseline (without failures). Under this attacker, single and double safeguard algorithms perform similarly.

(Parameters: $m = 10$, $\alpha = 0.4$. Single safe-guard threshold $\mathfrak{T} = 2$, window size $T = 3$ epochs, variance $\nu^2 = 10^{-8}$; double safe-guard thresholds $\mathfrak{T}_0 = 2, \mathfrak{T}_1 = 7$ and window sizes $T_0 = 1$ epoch, $T_1 = 6$ epochs, variance $\nu^2 = 10^{-8}$. Zeno Batch size $n_r = 4$ and constant weight $\rho = 0.0005$.)

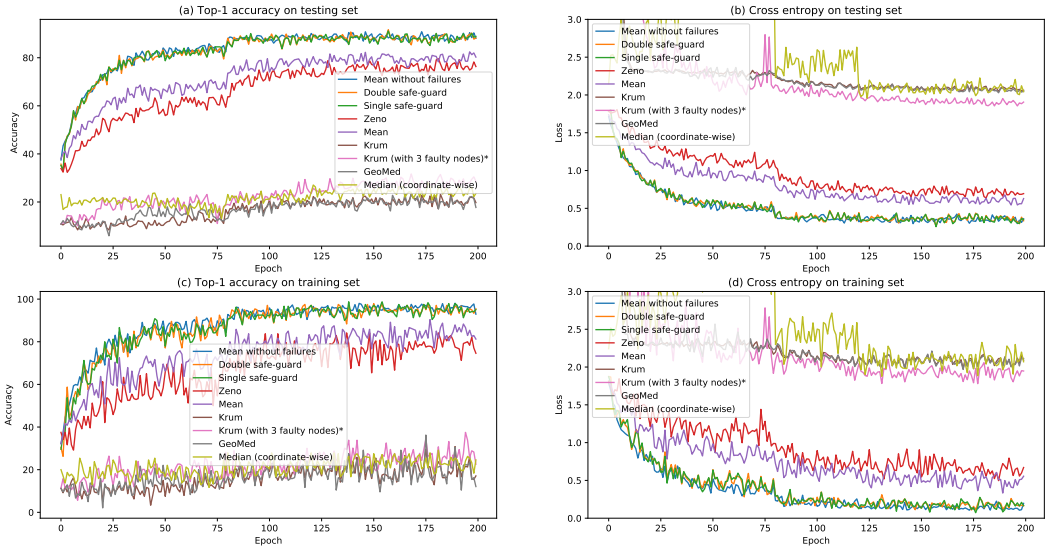


Figure 3: Comparison under delayed-gradient attack with delay $D = 1000$.

### D.2.3 Label-Flipping Attack

Recall that, in the label-flipping attack, each Byzantine worker computes its gradient based on the cross-entropy loss with flipped labels: for CIFAR-10, label $\ell \in \{0, ..., 9\}$ is flipped to $9 - \ell$.

The results are shown in Figure 4. From the plots, one can see that our double safe-guard algorithm outperforms prior works, and closely tracks the performance of the ideal baseline (without failures). Under this attacker, the double safeguard algorithm performs slightly better than single safeguard.

(Parameters: $m = 10$, $\alpha = 0.4$. Single safe-guard threshold $\mathfrak{T} = 0.4$, window size $T = 3$ epochs, variance $\nu^2 = 10^{-8}$; double safe-guard thresholds $\mathfrak{T}_0 = 2, \mathfrak{T}_1 = 7$ and window sizes $T_0 = 1$ epoch, $T_1 = 6$ epochs, variance $\nu^2 = 10^{-8}$. Zeno Batch size $n_r = 4$ and constant weight $\rho = 0.0005$.)

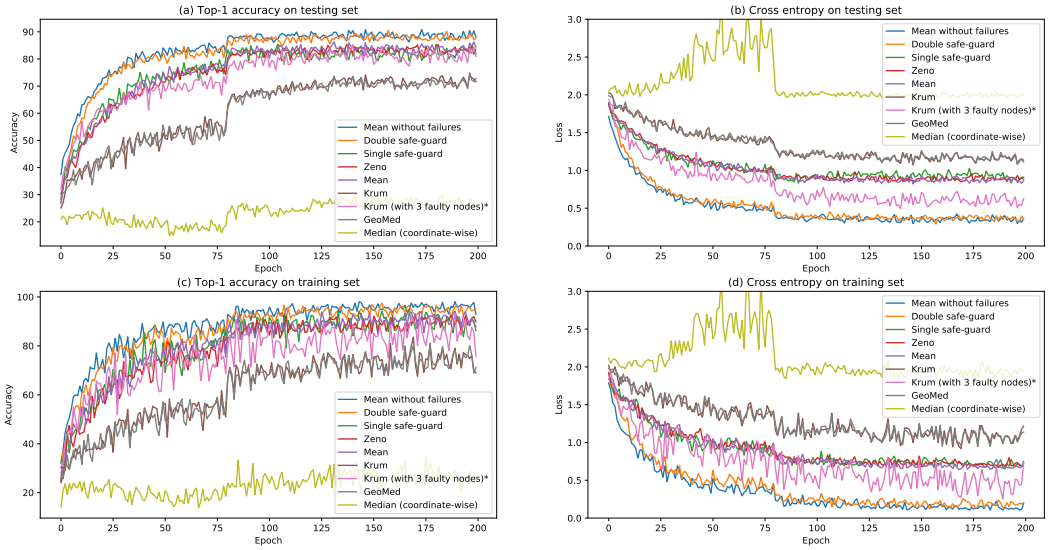


Figure 4: Comparison under label-flipping attack.

### D.2.4 Zero-Gradient Attack

Recall that, in the zero-gradient attack, each Byzantine worker sends zero gradient to the master. The result is shown in Figure 5. From the plots one can see that our single and double safe-guard algorithms, together with Zeno and Mean outperform other methods, and they closely track the performance of the ideal baseline

(without failures). It should not be surprising that the (Naive) Mean method performs well under zero-gradient attack: taking the naive mean now behaves equivalently as the perfect baseline with decreased learning rate.

(Parameters: $m = 10$, $\alpha = 0.4$. Single safe-guard threshold $\mathfrak{T} = 2$, window size $T = 3$ epochs, variance $\nu^2 = 10^{-8}$; double safe-guard thresholds $\mathfrak{T}_0 = 1, \mathfrak{T}_1 = 2$ and window sizes $T_0 = 1$ epoch, $T_1 = 6$ epochs, variance $\nu^2 = 10^{-8}$. Zeno Batch size $n_r = 4$ and constant weight $\rho = 0.0005$.)

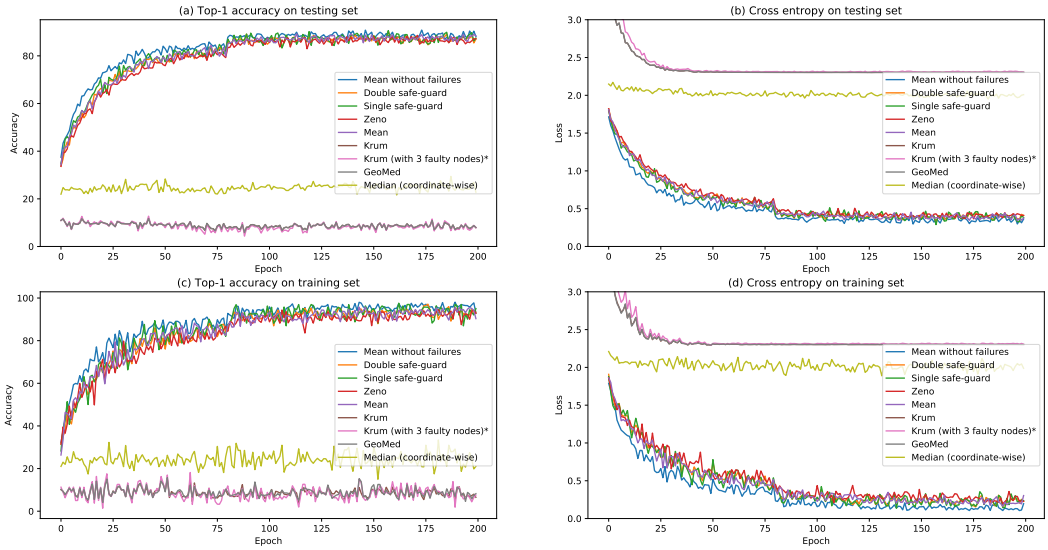


Figure 5: Convergence under zero-gradient attack.

### D.2.5 SAFEGUARD ATTACK

Recall that, in the safeguard attack, Byzantine workers send negative but re-scaled gradient to the master. We choose the re-scale factor so that it hardly triggers the safe-guard conditions at the master. From our experiment, choosing the re-scale factor as $0.4$ in most cases do not trigger the safe-guard conditions, while re-scale factor $0.5$ enables the algorithm to catch some Byzantine workers.

Our results are shown in Figure 7 (for re-scale factor $0.4$) and Figure 8 (for re-scale factor $0.5$). In Figure 7, the performance of our (single and double) safeguard algorithms indeed get impacted, but our algorithm still outperforms or matches all prior works. The performance of our (single and double) safeguard algorithms hardly get impacted, and closely tracks the performance of the ideal baseline (and outperforms other prior works significantly).

| | Defense | Best Test Accuracy | Last Test Accuracy | Best Loss | Last Loss |
|---|---|---|---|---|---|
| Safeguard attack (rescaling-factor = 0.5) | Mean without Failures | 90.8 | 88.1 | 0.30 | 0.36 |
| | **Double Safe-guard** | **91.4** | 88.5 | **0.26** | **0.36** |
| | Single Safe-guard | 90.8 | **88.8** | 0.27 | 0.36 |
| | Zeno | 86.5 | 84.8 | 0.45 | 0.50 |
| | Krum / Krum (α=0.3) | -/ 24.1 (α=0.3) | -/ 20.4 (α=0.3) | -/ > 3 (α=0.3) | -/ > 3 (α=0.3) |
| | GeoMed | 16.4 | 11.8 | > 3 | > 3 |
| | Coordinate-wise Median | 23.8 | 18.5 | 2.16 | 2.35 |

Figure 6: Full results under safe-guard attack with re-scale factor = 0.5.

(Figure 7 parameters: $m = 10$, $\alpha = 0.4$. Single safe-guard threshold $\mathfrak{T} = 2$, window size $T = 3$ epochs, variance $\nu^2 = 10^{-8}$; double safe-guard thresholds $\mathfrak{T}_0 = 1, \mathfrak{T}_1 = 2$ and window sizes $T_0 = 1$ epoch, $T_1 = 6$ epochs, variance $\nu^2 = 10^{-8}$. Zeno Batch size $n_r = 4$ and constant weight $\rho = 0.0005$.)

(Figure 8 parameters: $m = 10$, $\alpha = 0.4$. Single safe-guard threshold $\mathfrak{T} = 0.8$, window size $T = 3$ epochs, variance $\nu^2 = 10^{-8}$; double safe-guard thresholds $\mathfrak{T}_0 = 0.8, \mathfrak{T}_1 = 2$ and window sizes $T_0 = 1$ epoch, $T_1 = 6$ epochs, variance $\nu^2 = 10^{-8}$. Zeno Batch size $n_r = 4$ and constant weight $\rho = 0.0005$.)

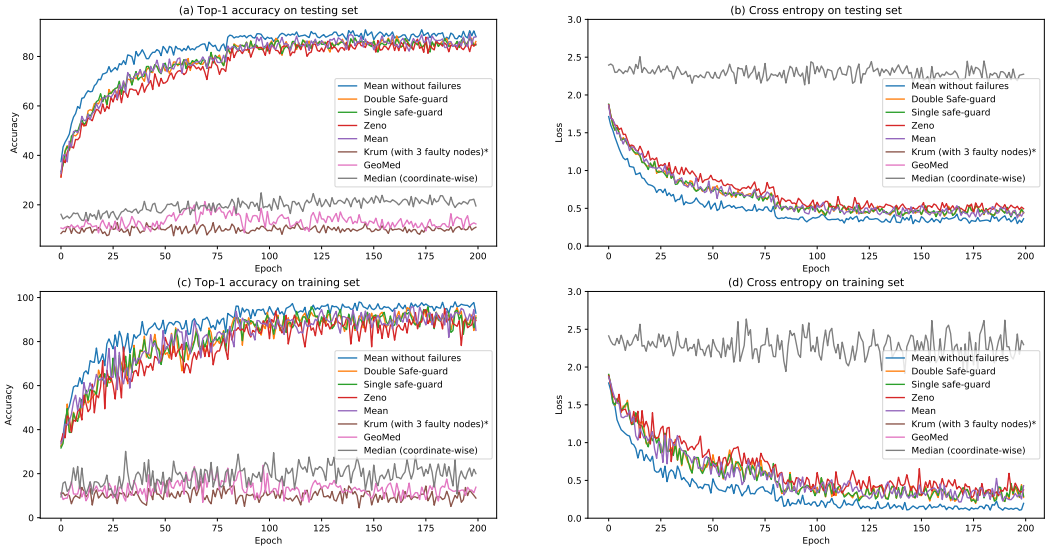


Figure 7: Comparison under safe-guard attack with re-scale factor = 0.4.

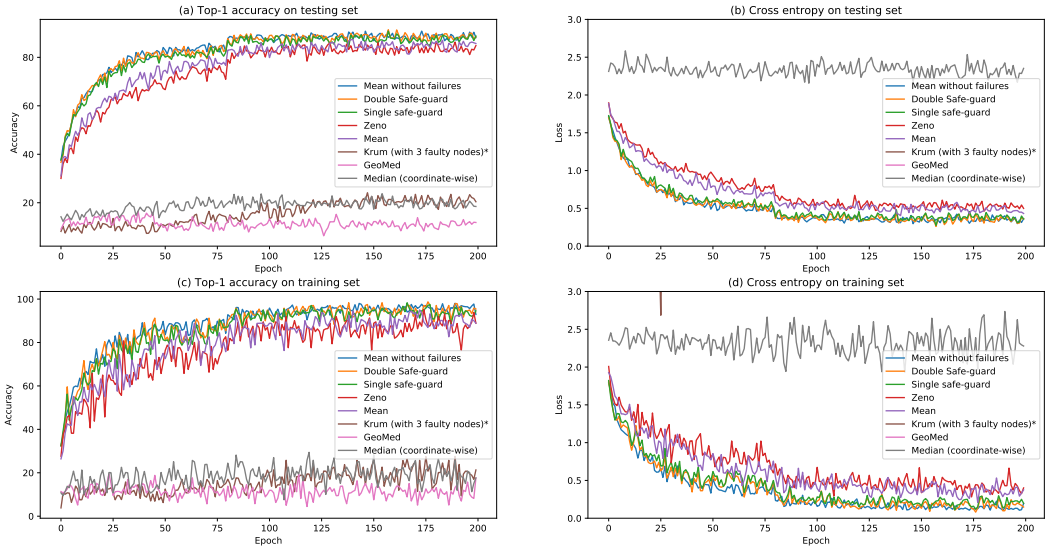


Figure 8: Convergence under safe-guard attack with re-scale factor = 0.5.

### D.3 Variance Attack

The Variance attack follows the strategy prescribed by Baruch et al. (2019), by which Byzantine workers collude in order to shift the mean among all gradients by a factor $\beta$ times the standard deviation of the gradient, while staying within population variance. More precisely, the maximal change to the mean that can be applied by an attacker without the fear of being detected, by using the properties of the Normal distribution, specifically the cumulative standard normal function, and compute the maximal possible shift so that the attackers's values stay within population variance. (Please see (Baruch et al., 2019, Algorithm 3) for a precise description. Our $\beta$ is called $z_{\max}$ in their notation.) We implement this strategy coordinate-wise, as in the original implementation of Baruch et al. (2019). Their work observes that the shift $\beta$ can be non-trivial in practice, since stochastic gradients tend to have large variance in neural network training (which we also observed in our setup). Critically, the attack cannot be defended against by *historyless algorithms*, as the attacker's values are statistically indistinguishable from a regular execution in a single iteration. In our setting, for 10 total nodes and $\alpha = 0.4$, $\beta$ is upper bounded by $0.3$ (following the classic tables for the cumulative normal). We ran the same attack in the setup from their paper (50 nodes total, of which 24 are Byzantine, which allows $\beta \sim 1.5$) and observed exactly the same outcome. Results for this experiment are given in Table 3, while the convergence of different schemes under this attack is presented in Figure 9.

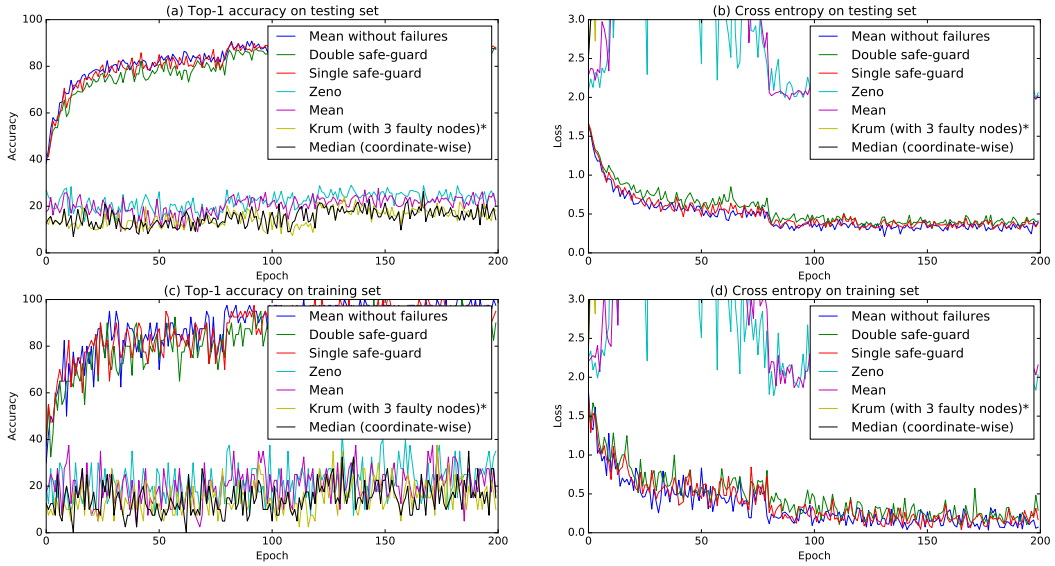


Figure 9: Convergence under variance attack with stdev factor = 0.3.

As shown by the results, our algorithm provably circumvents this attack, and recovers full accuracy. This is because both variants of the algorithm successfully tag the bad nodes after a few hundred iterations. This is explained by the fact that the algorithm has *memory*: in particular, Byzantine nodes following this strategy will progressively diverge from the (honest) "median" $B_{\text{med}}$ (at a "linear" rate), and therefore will eventually exceed the threshold and be marked as malicious by the algorithm. (In practice, the norm of their $B_i$ vector increases roughly monotonically until it surpasses the threshold.)

(Figure 9 parameters: $m = 10$, $\alpha = 0.4$. Single safe-guard threshold $\mathfrak{T} = 0.8$, window size $T = 3$ epochs, variance $\nu^2 = 10^{-8}$; double safe-guard thresholds $\mathfrak{T}_0 = 0.8, \mathfrak{T}_1 = 2$ and window sizes $T_0 = 1$ epoch, $T_1 = 6$ epochs, variance $\nu^2 = 10^{-8}$. Zeno Batch size $n_r = 4$ and constant weight $\rho = 0.0005$. Weight decay was enabled.)

### D.4 Full Comparison Table

Please see Table 3.

### D.5 Attack against the convex algorithm of Alistarh et al. (2018)

We now briefly describe an attack against this algorithm. The attack specifically leverages the fact that the algorithm does not use sliding windows.

One can first run the vanilla SGD to compute "the maximum deviation per good worker" for the accumulation vector used by the algorithm $\sum_{t=0}^{T} \nabla_t$. This maximum deviation is therefore a lower bound for the threshold used in their algorithm. Next, we design an attacker who evenly distributes this total allowed deviation to e.g. 5 consecutive epochs, and behaves honestly for the remaining epochs. Such an attacker cannot be identified by this algorithm, because its total deviation across all the iterations is identical to that of a good worker. However, this leads the algorithm to divergence.

Specifically, suppose 4 Byzantine workers all maliciously report their stochastic gradients multiplied by the scalar $-5$, and the remaining 6 good workers report their true stochastic gradients. One can verify numerically that this attacker can run for 5 consecutive epochs (say, epochs $a, a+1, a+2, a+3, a+4$) without being caught by the algorithm. Now,

- if $a \leq 75$, within just 1 epoch of attack, the neural net weights diverge (value NaN).
- if $80 \leq a \leq 115$, this attack is applied after the first learning rate decay. Within just 1 epoch of the attack, the objective explodes and accuracy becomes $10\%$ (random), and within 3 epochs the algorithm diverges completely.
- if $120 \leq a \leq 155$, this attack is after the second learning rate decay. Within just 2 epochs of attack, the accuracy drops to $11\%$. Later, the accuracy never recovers above $40\%$.

| Attack | Algorithm | Best Test Accuracy | Test Accuracy (Epoch 200) | Best Cross Entropy on Testing Set | Cross Entropy on Testing Set (Epoch 200) |
|---|---|---|---|---|---|
| Delayed-gradient | Mean without Failures | 90.8 | 88.1 | 0.30 | 0.36 |
| | **Double Safe-guard** | **91.7** | 88.2 | **0.26** | **0.33** |
| | Single Safe-guard | 91.1 | **88.9** | 0.26 | 0.35 |
| | Zeno | 79.6 | 76.3 | 0.64 | 0.69 |
| | Krum / Krum (α=0.3) | 23.4/ 31.7 (α=0.3) | 19.7/ 28.6 (α=0.3) | 2.04/ 1.84 (α=0.3) | 2.06/ 1.90 (α=0.3) |
| | GeoMed | 23.5 | 17.7 | 2.03 | 2.05 |
| | Coordinate-wise Median | 28.1 | 23.5 | 1.96 | 2.04 |
| Label-flipping | Mean without Failures | 90.8 | 88.1 | 0.30 | 0.36 |
| | **Double Safe-guard** | **90.6** | **87.8** | **0.29** | **0.36** |
| | Single Safe-guard | 85.5 | 83.6 | 0.83 | 0.88 |
| | Zeno | 85.3 | 82.0 | 0.85 | 0.91 |
| | Krum / Krum (α=0.3) | 74.1/ 85.3 (α=0.3) | 71.6/ 81.0 (α=0.3) | 1.07/ 0.50 (α=0.3) | 1.14/ 0.62 (α=0.3) |
| | GeoMed | 74.9 | 72.7 | 1.08 | 1.11 |
| | Coordinate-wise Median | 30.1 | 25.9 | 1.94 | 2.00 |
| Sign-flipping | Mean without Failures | 90.8 | 88.1 | 0.30 | 0.36 |
| | Double Safe-guard | 90.5 | 88.0 | 0.30 | 0.37 |
| | **Single Safe-guard** | **90.9** | **88.3** | **0.27** | **0.34** |
| | Zeno | 74.4 | 71.2 | 0.82 | 0.86 |
| | Krum / Krum (α=0.3) | 54.5/ 72.3 (α=0.3) | 53.2/ 67.2 (α=0.3) | 1.28/ 0.83 (α=0.3) | 1.33/ 0.96 (α=0.3) |
| | GeoMed | 55.5 | 53.8 | 1.25 | 1.30 |
| | Coordinate-wise Median | 22.7 | 19.4 | 2.20 | 2.36 |
| Zero-gradient | Mean without Failures | 90.8 | 88.1 | 0.30 | 0.36 |
| | Double Safe-guard | 90.3 | 87.0 | 0.31 | 0.41 |
| | **Single Safe-guard** | **90.6** | **87.5** | **0.29** | **0.38** |
| | Zeno | 88.6 | 87.2 | 0.36 | 0.41 |
| | Krum / Krum (α=0.3) | 11.8/ 12.7 (α=0.3) | 8.0/ 8.0 (α=0.3) | 2.30/ 2.30 (α=0.3) | 2.30/ 2.31 (α=0.3) |
| | GeoMed | 11.8 | 8.0 | 2.30 | 2.30 |
| | Coordinate-wise Median | 29.4 | 22.8 | 1.95 | 2.01 |
| Safeguard attack (rescaling-factor = 0.4) | Mean without Failures | 90.8 | 88.1 | 0.30 | 0.36 |
| | **Double Safe-guard** | **88.3** | **86.1** | **0.39** | **0.44** |
| | Single Safe-guard | 87.5 | 84.7 | 0.39 | 0.45 |
| | Zeno | 87.7 | 84.9 | 0.44 | 0.49 |
| | Krum / Krum (α=0.3) | -/ 12.8 (α=0.3) | -/ 10.9 (α=0.3) | -/ > 3 (α=0.3) | -/ > 3 (α=0.3) |
| | GeoMed | 21.3 | 12.2 | > 3 | > 3 |
| | Coordinate-wise Median | 24.9 | 19.5 | 2.13 | 2.27 |
| Variance attack (stdev-factor = 0.3) | Mean without Failures | 90.8 | 87.4 | 0.21 | 0.4 |
| | **Double Safe-guard** | **90.8** | **88.2** | **0.26** | **0.38** |
| | Single Safe-guard | 91.0 | 87.4 | 0.29 | 0.38 |
| | Zeno | 29.0 | 22.0 | 1.95 | 2.06 |
| | Krum (α=0.3) | 24.2 (α=0.3) | 14.2 (α=0.3) | 2.53 (α=0.3) | 2612.3 (α=0.3) |
| | Coordinate-wise Median | 26.4 | 14.2 | 2.51 | 12106.37 |

Table 3: Table of results for CIFAR10 dataset and ResNet20 model.