Bant: Byzantine Antidote via Trial Function and Trust Scores

Gleb Molodtsov[1,2,3,5*], Daniil Medyakov[1,2,3*], Sergey Skorik[4], Nikolas Khachaturov[6], Shahane Tigranyan[7], Vladimir Aletov[1,2,3], Aram Avetisyan[4], Martin Takáč[8], Aleksandr Beznosikov[1,2,3,5]

[1]Moscow Independent Research Institute of Artificial Intelligence
[2]Basic Research of Artificial Intelligence Laboratory (BRAIn Lab)
[3]Federated Learning Problems Laboratory
[4]Trusted AI Research Center, RAS
[5]Innopolis University
[6]Russian-Armenian University
[7]Institute for Informatics and Automation Problems, NAS RA
[8]Mohamed bin Zayed University of Artificial Intelligence
gleb.molodcov@gmail.com, medyakovd3@gmail.com

Abstract

Recent advancements in machine learning have improved performance while also increasing computational demands. While federated and distributed setups address these issues, their structures remain vulnerable to malicious influences. In this paper, we address a specific threat: Byzantine attacks, wherein compromised clients inject adversarial updates to derail global convergence. We combine the concept of trust scores with trial function methodology to dynamically filter outliers. Our methods address the critical limitations of previous approaches, allowing operation even when Byzantine nodes are in the majority. Moreover, our algorithms adapt to widely used scaled methods such as Adam and RMSProp, as well as practical scenarios, including local training and partial participation. We validate the robustness of our methods by conducting extensive experiments on both public datasets and private ECG data collected from medical institutions. Furthermore, we provide a broad theoretical analysis of our algorithms and their extensions to the aforementioned practical setups. The convergence guaranties of our methods are comparable to those of classical algorithms developed without Byzantine interference.

Code — https://github.com/brain-lab-research/Bant

1 Introduction

As the field of machine learning expands, researchers are confronted with challenges stemming from increasingly complex models and larger computational demands. To address these issues, distributed and federated learning scenarios have been developed (Kairouz et al. 2021; Konečný et al. 2016; Li et al. 2020). These approaches are crucial for a wide range of tasks (Smith et al. 2017; McMahan et al. 2017; Verbraeken et al. 2020), yet, they introduce several complications. Making the training process multi-node leads to threats related to data storage and transmission. These vulnerabilities manifest as device malfunctions, incorrect data relays, or even initial data corruption (Biggio, Nelson, and Laskov 2012). (Wang et al. 2023) demonstrated how adversarial attacks on data integrity compromise training. In the midst of them are Byzantine attacks. They occur in networks where certain workers, known as Byzantines, may corrupt data (Blanchard et al. 2017).

This paper specifically examines the threat of Byzantine attacks. We highlight the limitations of existing protection mechanisms, particularly their reliance on strong assumptions. In response, we propose an approach that leverages diverse concepts to develop a universal method, free from these constraints and applicable to practical scenarios.

Setup. We examine the problem often encountered in distributed machine learning, with $\mathcal{D}_i$ representing an unknown distribution of the training sample data on the i-th device:

$$\min_{x\in\mathbb{R}^d}\big[f(x)=\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{\xi_i\sim\mathcal{D}_i}\left[f_{\xi_i}(x)\right]\big]. \tag{1}$$

We consider a setup involving n workers connected to a central server. These workers are divided into two categories: Good (or Honest) workers, indicated by $\mathcal{G}$, and Byzantines, indicated by $\mathcal{B}$. At each iteration t, the sets $\mathcal{G}(t)$ and $\mathcal{B}(t)$ are redefined. This allows the composition of honest and Byzantine workers to vary dynamically over time. At every step, we assume that the set of honest workers $\mathcal{G}(t)$ is nonempty, i.e., $G(t) := |\mathcal{G}(t)| \geqslant 1$. During training, the number of Byzantines at each iteration remains unknown. This number is used only in the theoretical analysis of the worst-case scenario.

2 Related Work

Sophisticated attacks. Methods resistant to Byzantine attacks are crucial for solving optimization problems. Classical methods for distributed optimization, such as SGD (Robbins and Monro 1951; Bottou 2012; Recht and Ré 2013; McMahan et al. 2017), ADAM (Kingma and Ba 2014; Reddi et al. 2020), and SCAFFOLD (Karimireddy et al. 2020), average the received gradients or models. However, they *cease to operate when even a single Byzantine worker appears in the network* (Blanchard et al. 2017). Given the critical importance of this

*These authors contributed equally.

problem, numerous publications have addressed it (Feng, Xu, and Mannor 2014; Damaskinos et al. 2019). Initial approaches proposed robust aggregation rules for data from devices, such as COORDINATE-WISE MEDIAN, TRIMMED MEAN (Yin et al. 2018), KRUM (Blanchard et al. 2017), BULYAN (Mhamdi, Guerraoui, and Rouault 2018). However, sophisticated attacks, such as ALIE (Baruch, Baruch, and Goldberg 2019) or INNER PRODUCT MANIPULATION (Xie, Koyejo, and Gupta 2020), managed to *circumvent these aggregation rules* by shifting the mean they sought to find.

Clipping, Momentum, and Variance Reduction. Aggregation rules are *non-robust even in the absence of attacks*. For example, this occurs in cases of imbalanced classes. This issue was addressed in (Karimireddy, He, and Jaggi 2021b), where CENTERED CLIP (CC) technique was revealed. Additionally, the authors highlighted that the aforementioned Byzantine-robust methods cannot converge with any predetermined accuracy. Given the importance of this issue, they added client momentum, effectively addressing the problem. Another approach to combat the Byzantines is the application of the variance reduction technique. Originally introduced to eliminate irreducible errors in stochastic methods, it was subsequently proposed as an effective way to mitigate the presence of noise in computing stochastic gradient estimators (Gorbunov et al. 2023). Then this idea was developed in (Malinovsky et al. 2023).

These methods demonstrated significant improvements in resilience against Byzantine attacks. However, they also contain notable limitations. First, variance reduction methods exhibit *moderate convergence* in deep learning applications and are prone to *overfitting* (Defazio and Bottou 2019). Additionally, all aforementioned approaches *suffer from the requirement that the majority of devices must be honest*.

Validation tests. Another approach to achieving solutions with any specified accuracy involves techniques such as validation tests (Alistarh, Allen-Zhu, and Li 2018; Allen-Zhu et al. 2020) or computation checks (Gorbunov et al. 2022). Nevertheless, they still require a *majority of honest devices* and rely on *strict assumptions* in the analysis (Alistarh, Allen-Zhu, and Li 2018; Allen-Zhu et al. 2020).

Heterogeneous setup. While much work on the Byzantines focused on the distributed case with homogeneous data, a different series of papers allowed for data heterogeneity (Wu et al. 2020; El-Mhamdi et al. 2021; Data and Diggavi 2021; Nguyen et al. 2022). This corresponds, for example, to the federated setting. Methods were primarily built around robust aggregation (Karimireddy, He, and Jaggi 2021a; Chang et al. 2019; Data and Diggavi 2021; Allouah et al. 2024c; Dorfman, Yehya, and Levy 2024; Allouah et al. 2024b), and variance reduction techniques (Allouah et al. 2023). One of the most advanced approaches assigned coefficients (TRUST SCORES) to clients based on their reliability, using these scores to perform gradient steps (Cao et al. 2021; Yan et al. 2024). These studies provided a foundation for Byzantine-robust optimization in the federated setup, yet suffered from *the aforementioned drawbacks*.

Majority of attackers – trial function. To avoid requiring a majority of honest workers, several methods leverage server-held ground-truth data to filter compromised updates. ZENO (Xie, Koyejo, and Gupta 2019) implements this idea by maintaining a small validation set on the server and scoring each incoming gradient against the gradient computed on that set. We define the function evaluated on the validation set as the TRIAL FUNCTION. As for ZENO, it uses the trial function to compute trust scores but primarily aggregates updates through simple averaging, down-weighting or excluding devices with very low trust. As a result, its performance depends strongly on the devices that are treated as trusted. (Cao and Lai 2019) proposed an alternative that filters updates by comparing them to a noisy gradient approximation computed on a small dataset, which is effectively equivalent to using a validation set as in ZENO. The idea of using a small server-side dataset to validate local updates was also utilized in SAGEFLOW (Park et al. 2021), which handles majority adversarial clients. However, this method *performs poorly* under an attack ratio of 60%. Moreover, all aggregation schemes that discard clients based on trust scores remain *sensitive to hyperparameters*, as they require threshold choices to label devices as malicious. It requires tuning, and we discuss such instability regarding ZENO in Appendix, Table 7.

The authors of (Xie, Koyejo, and Gupta 2019; Cao and Lai 2019; Park et al. 2021) extended their results to address data heterogeneity by requiring the server to possess a representative sample of all device data. However, this assumption is *unrealistic* in real-world scenarios, undermining the fundamental achievements of federated learning with respect to privacy. Approaches (Guo et al. 2021, 2024) also *accumulate user data* on the server, raising doubts about applicability.

Other shortcomings of previous research. In addition to aforementioned limitations, many studies in this field are predominantly *heuristic and lack rigorous theoretical analysis* (Yan et al. 2024; Guo et al. 2021, 2024; Chang et al. 2019; Xu and Lyu 2020; Rodríguez-Barroso et al. 2022; Nguyen et al. 2022; Zhang et al. 2022; Huang et al. 2024). Moreover, in some studies, the *practical component seems flawed* due to the absence of experiments assessing test accuracy (Gorbunov et al. 2023). Furthermore, theoretical settings often do not align with practical aspects. For instance, in (Cao et al. 2021), *homogeneous data sampling is assumed* while focusing on the federated learning. Besides, the analysis of SAGEFLOW is confined to an *unrealistic strongly convex setup*.

Furthermore, when addressing problems in the distributed or federated setups, local methods (Woodworth, Patel, and Srebro 2020; Khaled, Mishchenko, and Richtárik 2020; Gorbunov, Hanzely, and Richtárik 2021; Nguyen et al. 2022), as well as the partial participation scenario (Yang, Fang, and Liu 2021; Kairouz et al. 2021; Sadiev et al. 2022; Nguyen et al. 2022), is typically assumed. While these options improve computational efficiency and reduce data transmission overhead, only a few studies (Data and Diggavi 2021; Malinovsky et al. 2023; Allouah et al. 2024a; Dorfman, Yehya, and Levy 2024) address these aspects, whereas the majority of works do not. In addition, research is often limited to SGD-like methods, *neglecting adaptive algorithms* such as ADAM (Kingma and Ba 2014) and RMSPROP (Tieleman and Hinton 2012), which are widely used in machine learning. Given the challenges and gaps identified in the existing lit-

erature, we aim to advance the trust score methodology and trial function concept to enhance defense mechanisms.

2.1 Contributions

Our main results are summarized as follows.

• **Combine trust scores with the trial function approach.** Trial loss is based on a subset of the training data stored on the server. Weights are assigned to the gradients sent from each device based on the extent to which these gradients reduce the trial function in each iteration. In real networks, honest stochastic gradients may increase the target loss. We account for this by incorporating weights from the previous epoch and a momentum parameter for more stable convergence.

• **Milder assumptions.** Unlike most existing studies, our approach requires only one reliably honest worker instead of a majority. Moreover, unlike previous trial function-based methods that assume data homogeneity (Cao et al. 2021; Gorbunov et al. 2022), our algorithms operate under the more realistic assumption of data similarity in federated learning.

• **Extensions.** We adapt our algorithms to important scenarios that are often overlooked in research.

(a) **Local methods.** In our work, we propose utilizing Local SGD to address the high communication costs typically associated with distributed training.

(b) **Partial participation.** Our algorithms incorporate the option for partial participation. Thus, devices may not participate in every learning step, and the attackers may vary across iterations.

(c) **Adaptive methods.** In this work, we extend our analysis to adaptive algorithms (e.g., ADAM and RMSPROP), which are widely used in machine learning.

• **Convergence guarantees.** We prove upper bounds on the convergence rates of the main methods and extensions presented for the smooth problem under various assumptions regarding the convexity of the target function (strong convexity, convexity, non-convexity).

• **Experiments.** We demonstrate the superiority of our method in both previously studied attacks and scenarios where alternative methods fail. Our experiments are performed on the CIFAR-10 dataset and real ECG data, utilizing RESNET-18 and RESNET-1D18 neural networks, respectively. Additionally, we validate our approach to Learning-to-Rank tasks by training a Transformer-based ranking model.

3 Methodology

To tackle Byzantine attacks, we introduce a pivotal component of our methodology — a trial loss function $\hat{f}$. In the homogeneous setting (1) with $\mathcal{D}_i = \mathcal{D}$, we take a separate sample from $\mathcal{D}$, but in a smaller volume than the entire dataset. The trial function calculated on this data forms $\hat{f}$. Under the heterogeneous data scenario, we sample from the distribution $\mathcal{D}_1$ on the server to obtain delayed data for $\hat{f}$ (indexing the server does not violate generality; we further consider it a device with index 1). Formally, we can write the trial function as $\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} f_1(x, \xi_i)$, where N is the number of samples in $\hat{f}$. This function is stored on the server (obviously, an honest device). The sample distribution of the trial function is similar to the entire distribution $\mathcal{D}$ due to the property of data similarity. Besides, in practical scenarios, a server may not be able to share the entire dataset, providing only a sample of size N. Depending on the size of this sample, f_1 may differ from $\hat{f}$. Nevertheless, the larger the volume of this delayed sample, the closer $\hat{f}$ approximates the function f_1 (discussed in Lemma C.1 in Appendix). A small public or synthetic trial dataset is a practical assumption in Byzantine-robust federated learning. Methods such as FLTrust (Cao et al. 2021) and Zeno (Xie, Koyejo, and Gupta 2019) utilize such datasets. Here we outline the assumptions under which we establish the convergence rates.

Assumption 3.1. The function $\hat{f}$ is L-smooth, i.e., $\|\nabla \hat{f}(x) - \nabla \hat{f}(y)\| \leqslant L\|x - y\|$ for any $x, y \in \mathbb{R}^d$.

Assumption 3.2. The function $\hat{f}$ is:

1. μ-**strongly convex** if for all $x, y \in \mathbb{R}^d$, it satisfies: $\hat{f}(y) \geqslant \hat{f}(x) + \langle \nabla \hat{f}(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2$.
2. **convex** if for all $x, y \in \mathbb{R}^d$, it satisfies: $\hat{f}(y) \geqslant \hat{f}(x) + \langle \nabla \hat{f}(x), y - x\rangle$.
3. **non-convex** if it has at least one (not necessarily unique) minimum, i.e., $\hat{f}(\hat{x}^*) = \inf\limits_{x \in \mathbb{R}^d} \hat{f}(x) > -\infty$.

Assumption 3.3. Each worker $i \in \mathcal{G}(t)$ has access to an independent and unbiased stochastic gradient with $\mathbb{E}[g_i(x, \xi_i)] = \nabla f_i(x)$ and its variance is bounded by σ^2:

$$\mathbb{E}\|g_i(x, \xi_i) - \nabla f_i(x)\|^2 \leqslant \sigma^2, \quad \text{for all } x \in \mathbb{R}^d.$$

We also assume data similarity – a common premise for ensuring convergence in Byzantine literature (Karimireddy, He, and Jaggi 2021a; Gorbunov et al. 2023; Yan et al. 2024).

Assumption 3.4. We assume data similarity in the following way: good clients possess (δ_1, δ_2)-heterogeneous local loss functions for some $\delta_1 \geqslant 0$ and $\delta_2 \geqslant 0$, such that for all $x \in \mathbb{R}^d$, the following holds:

$$\|\nabla f_i(x) - \nabla f(x)\|^2 \leqslant \delta_1 + \delta_2\|\nabla f(x)\|^2 \quad \forall i \in \mathcal{G}(t).$$

In over-parameterized models, introducing a positive δ_2 can sometimes reduce the value of δ_1. Several studies have explored heterogeneous scenarios in which honest workers handle distinct local functions (Wu et al. 2020; Karimireddy, He, and Jaggi 2021a). Note that achieving a predefined accuracy becomes feasible in the presence of Byzantines only when heterogeneity is limited to δ_2-bounded settings ($\delta_1 = 0$)(Karimireddy, He, and Jaggi 2021a). Under a more general assumption, which we utilize, the term with δ_1 inadvertently appears in the estimate.

Assumption 3.5. Byzantine workers are assumed to be omniscient, i.e., they have access to the computations performed by the other workers.

4 Algorithms and Convergence Analysis

4.1 First Method: `Bant`

In this section, we introduce our method, termed Byzantine ANTidote (`Bant`) – Algorithm 1. Our method relies on

Algorithm 1: Bant

1: **Input:** Starting point $x^0 \in \mathbb{R}^d$, $\omega_i^0 = 1/n \ \forall i$
2: **Parameters:** Stepsize $\gamma > 0$, momentum $\beta \in (0, 1]$
3: **for** $t = 0, 1, 2, \ldots, T-1$ **do**
4: Server sends x^t to each worker
5: **for all** workers $i = 1, 2, \ldots, n$ in parallel **do**
6: Generate ξ_i^t independently
7: Compute stochastic gradient $g_i^t = g_i(x^t, \xi_i^t)$
8: Send g_i^t to server
9: **end for**
10: $\omega_i^t = (1-\beta)\omega_i^{t-1} + \beta \frac{[\hat{f}(x^t) - \hat{f}(x^t - \gamma g_i^t)]_0}{\sum_{j=1}^n [\hat{f}(x^t) - \hat{f}(x^t - \gamma g_j^t)]_0}$
11: **if** each $[\hat{f}(x^t) - \hat{f}(x^t - \gamma g_i^t)]_0 = 0$ **then**
12: $\omega_i^t = (1-\beta)\omega_i^{t-1} + \beta \frac{1}{n}$
13: **end if**
14: $x^{t+1} = x^t - \gamma \sum_{i=1}^n \mathbb{I}_{[\hat{f}(x^t) - \hat{f}(x^t - \gamma g_i^t) > 0]} \omega_i^t g_i^t$
15: **end for**
16: **Output:** $\frac{1}{T} \sum_{t=0}^{T-1} x^t$

the core idea of assigning trust scores to devices. We integrate this with the concept of a trial function by aggregating the stochastic gradients g_i^t of devices with their respective weights w_i^t. To find the latter, we first calculate the contribution coefficients for each worker i at each step: $\theta_i^t = \hat{f}(x^t) - \hat{f}(x^t - \gamma g_i^t)$. These coefficients demonstrate how the i-th device affects convergence. If $\theta_i^t > 0$, the stochastic gradient minimizes trial loss and is assigned a weight. Otherwise, it is assigned a weight of zero (Line 10 in Algorithm 1). We ensure non-negativity with $[\theta_i^t]_0 = \max\{\theta_i^t, 0\}$ and normalize to provide a total weight of 1.

To address stochastic gradient instability, which can increase the loss function, we introduce a momentum parameter for the weights (Line 10). If all gradients increase the loss, they are assigned zero weights, thereby stopping the minimization process even in the absence of Byzantine devices in the network.

By adding momentum, we achieve a more stable convergence in practice. This allows previous favorable gradients to influence current weights, even if a device receives a small or zero weight in the current iteration. An indicator in the step (Line 14) ensures that gradients maximizing trial loss are ignored, thereby guaranteeing minimization at each step. We also define $G = \min_t G(t)$ as the minimum, taken over all iterations, of the number of honest workers at each iteration. Now we are ready to proceed with the theoretical results.

Theorem 4.1. *Under Assumptions 3.1, 3.2(2), 3.3, 3.4 with $\delta_2 \leqslant \frac{1}{12}$, 3.5, for solving the problem* (1)*, after T iterations of Algorithm 1 with $\gamma \leqslant \frac{1}{13L}$, the following holds:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(x^t)\|^2 \leqslant \frac{\mathbb{E}\left[\hat{f}(x^0) - \hat{f}(\hat{x}^*)\right]}{\gamma T} \cdot \frac{4n}{\beta G} + 6L\gamma\sigma^2 + 3\delta_1 + 4\zeta(N).$$

The first two terms in the result of Theorem 4.1 replicate the findings from the standard SGD analysis, up to constant factors. The last term, which depends on $\zeta(N)$, is of special interest. The function $\zeta(N)$ reflects the relationship between f_1 and the trial loss $\hat{f}$ and represents an approximation error. The dependence of $\zeta(N)$ on N is natural: the larger N, the smaller the error becomes. More precisely, for our function f, this error is $\zeta(N) = \widetilde{\mathcal{O}}\left(\frac{1}{N}\right)$ (see Lemma C.1 in Appendix). Although this error degrades convergence, it is common in machine learning tasks. In particular, the original learning problem, like (1), is often replaced by its Monte Carlo approximation (Johnson and Zhang 2013; Defazio, Bach, and Lacoste-Julien 2014; Allen-Zhu 2018), and the resulting problem is commonly referred to as empirical risk minimization (Shalev-Shwartz and Ben-David 2014). This replacement also leads to an error. Finally, δ_1 is a typical term that represents data similarity (Assumption 3.4) and is unavoidable in the presence of Byzantines (Wu et al. 2020; Karimireddy, He, and Jaggi 2021a; Gorbunov et al. 2022).

Since our approach resembles ZENO (Xie, Koyejo, and Gupta 2019) in its use of the trial function, we should mention that we found some issues in their proofs. In Theorem 1 of (Xie, Koyejo, and Gupta 2019), the authors incorrectly apply the expectation operator when deriving their recursion. They sample a trial function from the full dataset and use $\mathbb{E}[\hat{f}^t(x)] = f(x)$, which is valid for a random point. However, in the case of point x^{t+1}, a mistake was made. Since they sample $\hat{f}^t$ in every iteration, the point x^{t+1} depends on the sample $\hat{f}^t$, leading to $\mathbb{E}[\hat{f}^t(x^{t+1}) \mid x^t] \neq f(x^{t+1})$. By carrying the conditional expectation without considering the full expectation, it becomes impossible to enter the recursion and achieve convergence regarding the function f itself. In turn, we explicitly bound the gradient discrepancy $|\nabla f_1(x^t) - \nabla \hat{f}(x^t)|$, leveraging trial function sampling to ensure convergence as the sample size increases. This difference in Theorem 4.1 is represented by the discussed $\zeta(N)$.

Corollary 4.2. *Under the assumptions of Theorem 4.1, for solving the problem* (1)*, after T iterations of Algorithm 1 with $\gamma \leqslant \min\left\{\frac{1}{13L}, \frac{\sqrt{2\mathbb{E}\left[\hat{f}(x^0) - \hat{f}(\hat{x}^*)\right]n}}{\sigma\sqrt{3LG\beta T}}\right\}$, the following holds:*

$$\frac{1}{T}\sum_{t=1}^{T-1}\mathbb{E}\|\nabla f(x^t)\|^2 = \mathcal{O}\left(\frac{\mathbb{E}\left[\hat{f}(x^0) - \hat{f}(\hat{x}^*)\right]Ln}{\beta GT} + \frac{\sigma \cdot \sqrt{\mathbb{E}\left[\hat{f}(x^0) - \hat{f}(\hat{x}^*)\right] \cdot Ln}}{\sqrt{\beta GT}} + \delta_1 + \zeta(N)\right).$$

If we consider the first two terms in the convergence estimates from Corollary 4.2, the only difference from the classical SGD convergence results (Moulines and Bach 2011; Stich 2019) is the additional factor $\frac{n}{G}$, however, the rate is asymptotically optimal. The proofs and results for the strongly-convex objective can be found in Appendix.

4.2 Improving Theoretical Estimates: AutoBant

Despite practical advantages of Bant, it has some theoretical imperfections related to the mechanism of assigning trust scores. While parameter β helps honest clients maintain trust scores despite occasional bad gradients, it also enables Byzantine devices to retain their weights during attacks. To combat this, we add an indicator for the trial function reduction (the

Algorithm 2: AutoBant

1: **Input:** Starting point $x^0 \in \mathbb{R}^d$
2: **Parameters:** Stepsize $\gamma > 0$, error accuracy δ
3: **for** $t = 0, 1, 2, \ldots, T-1$ **do**
4: Server sends x^t to each worker
5: **for all** workers $i = 0, 1, 2, \ldots, n$ in parallel **do**
6: Generate ξ_i^t independently
7: Compute stochastic gradient $g_i^t = g_i(x^t, \xi_i^t)$
8: Send g_i^t to server
9: **end for**
10: $\omega^t \approx \arg\min_{\omega \in \Delta_1^n} \hat{f}\left(x^t - \gamma \sum_{i=1}^n \omega_i g_i^t\right)$
11: $x^{t+1} = x^t - \gamma \sum_{i=1}^n \omega_i^t g_i^t$
12: **end for**
13: **Output:** $\frac{1}{T}\sum_{t=0}^{T-1} x^t$

indication of the device being Byzantine at the considered iteration). However, this limits the theoretical applicability of the method to non-convex problems prevalent in modern machine learning. To resolve these limitations, we present our second method, called AUxiliary Trial Optimization for Byzantines ANTidote (AutoBant), Algorithm 2.

The idea of assigning weights to devices as part of the optimization process has gained popularity in federated learning. For instance, in many works, it leads to improved solution quality (Li et al. 2023; Tupitsa et al. 2024), or it is used in more specific settings, such as personalized learning (Mishchenko et al. 2023). We propose adapting this to Byzantine-robust learning by optimizing $\hat{f}$ regarding weights calculated after each algorithmic step (Line 10).

To solve the minimization problem, we can use various methods, e.g., Mirror Descent (Beck and Teboulle 2003; Allen-Zhu and Orecchia 2014):

$$\omega^{k+1} = \arg\min_{\omega \in \Delta_1^n} \left\{ \eta \left\langle \nabla_\omega \hat{f}\left(x^t - \gamma \sum_{i=1}^n \omega_i^k g_i^t\right), \omega \right\rangle + \mathcal{KL}(\omega \| \omega^k) \right\},$$

where $\mathcal{KL}(\cdot\|\cdot)$ denotes the Kullback-Leibler divergence. The error in solving this is bounded by δ:

$$\left| \min_{\omega \in \Delta_1^n} \hat{f}\left(x^t - \gamma \sum_{i=1}^n \omega_i g_i^t\right) - \hat{f}\left(x^t - \gamma \sum_{i=1}^n \omega_i^t g_i^t\right) \right| \leqslant \delta.$$

After solving this auxiliary problem, we produce an actual model update using the optimized weights (Line 11). In light of the proposed method, the question of the cost of implementing such an optimal scheme comes to the forefront. Note that the computational complexity of solving this subproblem at each iteration is only $\mathcal{O}(\log n/\delta^2)$ (Beck and Teboulle 2003), which is not critical. For the following theoretical analysis, we assume the minimization subproblem can be solved to arbitrary precision, neglecting errors in our estimates. In practice, however, the convergence rate is directly influenced by the accuracy of this solution. A numerical study of this effect is presented in Table 6 in Appendix. Now we are ready to present the main theoretical result of this section.

Theorem 4.3. *Under Assumptions 3.1, 3.2(3), 3.3, 3.4 with $\delta_2 < \frac{1}{12}$, 3.5, for solving the problem 1, after T iterations of Algorithm 2 with $\gamma \leqslant \frac{1}{13L}$, the following holds:*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(x^t)\|^2 \leqslant \frac{4\mathbb{E}\left[\hat{f}(x^0) - \hat{f}(\hat{x}^*)\right]}{\gamma T} + 3\delta_1 + \frac{6L\gamma}{G}\sigma^2 + 2\zeta(N).$$

Corollary 4.4. *Under assumptions of Theorem 4.3, for solving the problem* (1)*, after T iterations of Algorithm 2 with $\gamma \leqslant \min\left\{\frac{1}{13L}, \frac{\sqrt{2\mathbb{E}\left[\hat{f}(x^0) - \hat{f}(\hat{x}^*)\right]G}}{\sigma\sqrt{3LT}}\right\}$, the following holds:*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left\|\nabla f(x^t)\right\|^2 = \mathcal{O}\left(\frac{\mathbb{E}\left[\hat{f}(x^0) - \hat{f}(\hat{x}^*)\right]L}{T} + \delta_1 + \zeta(N) + \frac{\sigma\sqrt{\mathbb{E}\left[\hat{f}(x^0) - \hat{f}(\hat{x}^*)\right]L}}{\sqrt{TG}}\right).$$

Detailed proofs are presented in Appendix. In the first and second terms, we observe that the method converges in the same pace as the standard SGD only with honest workers (Ghadimi and Lan 2013; Ghadimi, Lan, and Zhang 2016). It turns out that we throw out all Byzantines, and this result is nearly optimal and unimprovable. As in Algorithm 1, a term responsible for the approximation error $\zeta(N)$ appears. Compared with the result of Corollary 4.2, we improve the rate through a more advanced aggregation mechanism. We remove the factor $\frac{n}{\beta G}$ from the main term and achieve a decrease in variance by a factor of G. However, an additional error δ is incurred, which can be viewed as a trade-off for solving the subproblem. Furthermore, our approach applies to a broader class of non-convex functions. Addressing the dependence of the stepsize on the number of Byzantines, this choice is based on the theoretical analysis of the worst-case scenario, considering the number of Byzantines. If this number is unknown, setting the minimum possible value of 1 eliminates this dependency.

5 Extensions

Byzantine robust optimization, as discussed earlier, lacks a solid theoretical foundation in several real-world settings. We address this gap. This section provides a brief overview of the scenarios to which we extend our analysis.

5.1 Local Methods

The main idea is that each device performs a predefined number of local steps. Then the aggregation of gradients and the mutual updates of model parameters, initialized by the server, take place. This reduces the number of communication rounds. However, it affects the convergence proportionally to the length of the communication round. Complete updates are performed only at specific iterations: $t = t_{k \cdot l}$ for some $k = \overline{0, \lfloor T/l \rfloor}$. During the remaining iterations, we perform local updates using the rule $x_i^{t+1} = x_i^t - \gamma g_i^t$. This approach ensures that communication overhead is minimized while maintaining efficient convergence.

Algorithm 3: `Scaled AutoBant` (part)

10: $\omega^t \approx \arg\min_{\omega \in \Delta_1^n} \hat{f}\left(x^t - \gamma \left(\hat{P}^t\right)^{-1} \sum_{i=1}^n \omega_i g_i^t\right)$

11: $x^{t+1} = x^t - \gamma \left(\hat{P}^t\right)^{-1} \sum_{i=1}^n \omega_i^t g_i^t$

Algorithm 4: `SimBant` (part)

10: $\omega_i^t = (1-\beta)\omega_i^{t-1} + \beta \frac{sim(m(x^t - \gamma g_i^t, \hat{\mathcal{D}}), m(x^t - \gamma g_1^t, \hat{\mathcal{D}}))}{\sum_{j=1}^n sim(m(x^t - \gamma g_j^t, \hat{\mathcal{D}}), m(x^t - \gamma g_1^t, \hat{\mathcal{D}}))}$

11: $x^{t+1} = x^t - \gamma \sum_{i=1}^n \omega_i^t g_i^t$

5.2 Partial Participation

It occurs when only a subset of clients actively participates in the training process during each communication round (Yang, Fang, and Liu 2021), allowing clients to join or leave the system. This approach is beneficial in scenarios such as mobile edge computing. However, it poses challenges such as incomplete model updates and potential degradation in model performance due to missed contributions from inactive clients (Wang and Ji 2022; Li et al. 2022). Our methods adapt to the scenario of partial participation by assigning trust scores to devices explicitly participating in training during the considered iteration. Furthermore, it is crucial to account for the minimum number of nodes participating in training across all iterations. Specifically, we analyze $\widetilde{G}(t) = \min_{t \leqslant T} G(t)$, where $G(t)$ denotes the set of active honest workers at iteration t.

5.3 Scaled Methods

Adaptive methods such as ADAM (Kingma and Ba 2014) and RMSPROP (Tieleman and Hinton 2012) have become widely popular due to their superior performance compared to standard SGD-like methods. We propose corresponding methods that utilize a diagonal preconditioner $(\hat{P}^t)^{-1}$, which scales a gradient to $(\hat{P}^t)^{-1} g_i^t$, and the step is executed using this scaled gradient. We present the part of `Scaled AutoBant` based on Algorithm 2. The estimates obtained are identical to those derived from the scaled methods in the non-Byzantine regime. All details can be found in Appendix.

5.4 Finding Scores from Validation

Another interesting direction to obtain trust scores w_i is to calculate *similarity between the logits* obtained on the server and on the device. The trust score for the i-th device is the function $\alpha_i \to sim(m(x^t - \gamma g_i^t, \hat{\mathcal{D}}), m(x^t - \gamma g_1^t, \hat{\mathcal{D}}))$. Based on Algorithm 2, we present a part of the `SimBant` algorithm (see details in Appendix).

6 Experiments

To evaluate the performance of the proposed methods, we conduct experiments on several benchmarks.

- **Classification Task:** We first validate our approach on the public dataset. We use RESNET-18 (He et al. 2016) for CIFAR-10 (Krizhevsky, Hinton et al. 2009) classification.
- **ECG Abnormality Detection:** In multi-hospital collaborations, labels are derived from expert annotations and automated pipelines, making attacks and subtle manipulations practical threats that can compromise patient safety. We obtain a proprietary dataset of 12-lead digital electrocardiograms (ECG) from five hospitals and train RESNET1D18 model for ECG abnormality detection.
- **Learning-to-Rank (Recommender Systems):** We conducted a series of experiments applied to the Learning-to-Rank (LTR) task, common in information retrieval and recommendation systems. We adopt the Transformer architecture (Vaswani et al. 2017), evaluating its performance on the dataset WEB30K (Qin and Liu 2013) under attacks.

We consider various Byzantine attacks to test our methods.
• **Label Flipping.** Attackers send gradients based on the loss calculated with randomly flipped labels.
• **Sign Flipping.** Attackers send the opposite gradient.
• **Random Gradients.** Attackers send random gradients.
• **IPM (Inner Product Manipulation).** Attackers send the average gradient of all honest clients multiplied by a factor of -κ (we set κ to 0.5) (Xie, Koyejo, and Gupta 2020).
• **ALIE (A Little Is Enough).** Attackers average their gradients and scale the standard deviation to mimic the majority (Baruch, Baruch, and Goldberg 2019).
We define the number of Byzantine clients as a percentage of the total number of clients, and specify it in the attack name. We train `Bant` and `AutoBant` in the scaled version (see Section 5.3) with the ADAM preconditioner. We include `SimBant`, ADAM, and the existing methods: ZENO (Xie, Koyejo, and Gupta 2019), RECESS (Yan et al. 2024), CENTERED CLIP (Karimireddy, He, and Jaggi 2021b), SAFEGUARD (Allen-Zhu et al. 2020), VR MARINA (Gorbunov et al. 2023) and FLTRUST (Cao et al. 2021). For CENTERED CLIP, we added techniques FIXING BY MIXING (Allouah et al. 2023) and BUCKETING (Karimireddy, He, and Jaggi 2021a). The methods were trained on the CIFAR-10 and ECG datasets for 200 and 150 rounds, respectively.

CIFAR-10 and ECG setups. For the CIFAR-10 dataset, we divide the data among 10 and 100 (see Appendix) clients. We consider a homogeneous split with 5,000 images per client, as well as a Dirichlet split with $\alpha = 0.5$ and $\alpha = 1$. We use 500 separate samples to form $\hat{f}$. For the ECG dataset, we consider five clients, each representing a hospital with 10,000 and 20,000 records. To form $\hat{f}$ on ECG, we use 100 samples from the publicly-available external PTB-XL dataset (Wagner et al. 2020). We solve the task of multiclass classification for CIFAR-10 and binary classification of 4 heart abnormalities for ECG: Atrial FIBrillation (AFIB), First-degree AV Block (1AVB), Premature Ventricular Complex (PVC), and Complete Left Bundle Branch Block (CLBBB).
The accuracy for all considered attacks on the CIFAR-10 test dataset are illustrated in Figure 1. To further stress test the proposed methods, we consider the most strong Byzantine attacks under heterogeneous setups, as well as homogeneous split under 100 clients. Figure 2 shows the accuracy plots of the proposed methods with Dirichlet $\alpha = 0.5$ for the ALIE and Random Gradients attacks. Details are in Appendix.

Algorithm	Without Attack		Label Flipping (60%)		Random Gradients (60%)		IPM (80%)		ALIE (40%)	
	G-mean	f1-score	G-mean	f1-score	G-mean	f1-score	G-mean	f1-score	G-mean	f1-score
ADAM	0.956±0.017	0.811±0.016	0.262±0.023	0.041±0.019	0.348±0.011	0.126±0.016	0.197±0.027	0.036±0.015	0.125±0.011	0.123±0.020
FLTRUST	0.952±0.020	0.800±0.019	0.952±0.016	0.753±0.011	0.617±0.020	0.174±0.019	0.061±0.017	0.125±0.015	0.017±0.013	0.123±0.018
RECESS	0.949±0.016	0.783±0.019	0.366±0.019	0.128±0.020	0.593±0.020	0.163±0.020	0.493±0.019	0.112±0.015	0.450±0.014	0.127±0.018
ZENO	0.921±0.012	0.787±0.014	0.014±0.017	0.110±0.015	0.163±0.010	0.089±0.014	0.102±0.012	0.066±0.018	0.010±0.009	0.091±0.011
CC	0.949±0.020	0.772±0.019	0.285±0.018	0.114±0.020	0.580±0.019	0.155±0.020	0.084±0.019	0.014±0.020	0.530±0.018	0.154±0.020
CC+fbm	0.954±0.016	0.808±0.020	0.840±0.019	0.716±0.014	0.562±0.011	0.151±0.020	0.027±0.018	0.123±0.015	0.876±0.017	0.594±0.013
CC+bucketing	0.947±0.013	0.790±0.018	0.829±0.011	0.708±0.020	0.570±0.012	0.164±0.018	0.035±0.020	0.118±0.012	0.870±0.019	0.587±0.014
SAFEGUARD	**0.957±0.020**	0.821±0.019	0.107±0.012	0.123±0.020	0.258±0.011	0.124±0.019	0.951±0.018	0.082±0.020	0.010±0.009	0.123±0.012
VR MARINA	0.010±0.014	0.120±0.010	0.027±0.018	0.123±0.020	0.176±0.012	0.103±0.013	0.127±0.013	0.079±0.019	0.012±0.010	0.108±0.013
Bant	0.953±0.017	**0.830±0.020**	**0.956±0.016**	**0.777±0.020**	**0.948±0.018**	**0.809±0.020**	0.946±0.020	0.676±0.015	**0.947±0.018**	**0.770±0.020**
AutoBant	0.953±0.019	0.781±0.020	0.790±0.020	0.276±0.020	0.946±0.019	0.748±0.018	0.942±0.020	0.690±0.020	0.892±0.016	0.585±0.020
SimBant	0.956±0.020	0.790±0.018	0.949±0.020	0.774±0.020	0.945±0.020	0.712±0.018	**0.955±0.020**	**0.783±0.018**	0.946±0.019	0.705±0.020

Table 1: RESNET1D18 on ECG (AFIB) for Byzantine-tolerance techniques under various attacks.

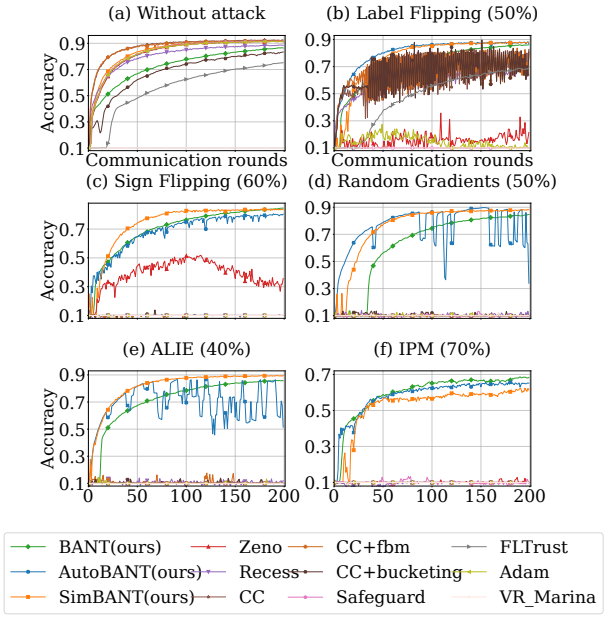

Figure 1: Test accuracy, ResNet18 on CIFAR-10.

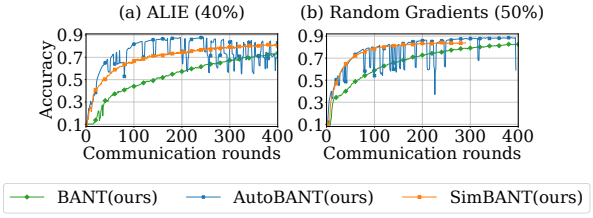

Figure 2: Test accuracy, ResNet18 on Dirichlet.

To assess model performance on the ECG data, we use the G-mean (the square root of sensitivity multiplied by specificity) and the f1-score metrics. Table 1 summarizes the results for the AFIB disease classification. Detailed results for other abnormalities are presented in Tables 9-12 in Appendix.
Unlike previous techniques, our methods exhibit robustness against all Byzantine attacks across different benchmarks. We note that `AutoBant` performs slightly worse than `Bant` and `SimBant` under Random Gradients and ALIE attacks. This occurs due to solving an auxiliary subproblem (Line 10 in Algorithm 2) using MIRROR DESCENT with KL-divergence. According to its properties, the algorithm assigns small but non-zero weights to Byzantines, contributing to unstable convergence, while `Bant` and `SimBant` lack this drawback. We analyze the required number of such iterations and the number of samples in the trial function in Appendix. RECESS and FLTRUST leverage the concept of trust scores but rely on a majority of honest devices. As a result, this leads to a significant decrease in the final quality under the majority of Byzantines in Random Gradients, IPM, and ALIE attacks that simulate a malicious majority. Similar behavior is observed for the CC and SAFEGUARD methods, which suffer from sensitivity to parameter tuning. FIXING BY MIXING and BUCKETING increase Label Flipping and ALIE metrics for ECG setup, but do not provide reliable convergence for all cases. ZENO exploits the trial function approach, however, it relies on the number of Byzantines (see Table 7 in Appendix).

Learning-to-Rank. In the LTR task, the goal is to learn a ranking function over query-document pairs. Each pair is represented using standard frequency-based feature vectors. The target labels correspond to human-assigned relevance scores. This setting provides a natural context for exploring Byzantine robustness. Annotators may provide inconsistent or biased relevance assessments. This reflects real-world challenges in supervised learning from human-generated data. We compare our methods against the baselines from prior experiments – under the most severe attacks, see Figure 3.

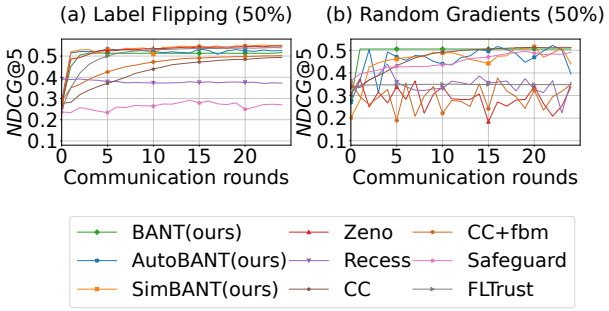

Figure 3: Test NDCG@5, Transformer on LTR task.

Acknowledgments

The work of Sergey Skorik and Aram Avetisyan was supported by a grant, provided by the Ministry of Economic Development of the Russian Federation (agreement dated June 20, 2025 No. 139-15-2025-011, identifier 000000C313925P4G0002).

References

Alistarh, D.; Allen-Zhu, Z.; and Li, J. 2018. Byzantine stochastic gradient descent. *Advances in neural information processing systems*, 31.

Allen-Zhu, Z. 2018. Katyusha: The first direct acceleration of stochastic gradient methods. *Journal of Machine Learning Research*, 18(221): 1–51.

Allen-Zhu, Z.; Ebrahimianghazani, F.; Li, J.; and Alistarh, D. 2020. Byzantine-Resilient Non-Convex Stochastic Gradient Descent. In *International Conference on Learning Representations*.

Allen-Zhu, Z.; and Orecchia, L. 2014. Linear coupling: An ultimate unification of gradient and mirror descent. *arXiv preprint arXiv:1407.1537*.

Allouah, Y.; Farhadkhani, S.; Guerraoui, R.; Gupta, N.; Pinot, R.; Rizk, G.; and Voitovych, S. 2024a. Byzantine-Robust Federated Learning: Impact of Client Subsampling and Local Updates. In *Forty-first International Conference on Machine Learning*. PMLR.

Allouah, Y.; Farhadkhani, S.; Guerraoui, R.; Gupta, N.; Pinot, R.; and Stephan, J. 2023. Fixing by mixing: A recipe for optimal byzantine ml under heterogeneity. In *International Conference on Artificial Intelligence and Statistics*, 1232–1300. PMLR.

Allouah, Y.; Guerraoui, R.; Gupta, N.; Jellouli, A.; Rizk, G.; and Stephan, J. 2024b. Boosting Robustness by Clipping Gradients in Distributed Learning. *arXiv preprint arXiv:2405.14432*.

Allouah, Y.; Guerraoui, R.; Gupta, N.; Pinot, R.; and Rizk, G. 2024c. Robust distributed learning: tight error bounds and breakdown point under data heterogeneity. *Advances in Neural Information Processing Systems*, 36.

Baruch, G.; Baruch, M.; and Goldberg, Y. 2019. A little is enough: Circumventing defenses for distributed learning. *Advances in Neural Information Processing Systems*, 32.

Beck, A.; and Teboulle, M. 2003. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3): 167–175.

Biggio, B.; Nelson, B.; and Laskov, P. 2012. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*.

Blanchard, P.; El Mhamdi, E. M.; Guerraoui, R.; and Stainer, J. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30.

Bottou, L. 2012. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade: Second Edition*, 421–436. Springer.

Cao, X.; Fang, M.; Liu, J.; and Gong, N. Z. 2021. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. In *ISOC Network and Distributed System Security Symposium (NDSS)*.

Cao, X.; and Lai, L. 2019. Distributed gradient descent algorithm robust to an arbitrary number of byzantine attackers. *IEEE Transactions on Signal Processing*, 67(22): 5850–5864.

Chang, H.; Shejwalkar, V.; Shokri, R.; and Houmansadr, A. 2019. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv preprint arXiv:1912.11279*.

Damaskinos, G.; El-Mhamdi, E.-M.; Guerraoui, R.; Guirguis, A.; and Rouault, S. 2019. Aggregathor: Byzantine machine learning via robust gradient aggregation. *Proceedings of Machine Learning and Systems*, 1: 81–106.

Data, D.; and Diggavi, S. 2021. Byzantine-resilient high-dimensional SGD with local iterations on heterogeneous data. In *International Conference on Machine Learning*, 2478–2488. PMLR.

Defazio, A.; Bach, F.; and Lacoste-Julien, S. 2014. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27.

Defazio, A.; and Bottou, L. 2019. On the ineffectiveness of variance reduced optimization for deep learning. *Advances in Neural Information Processing Systems*, 32.

Dorfman, R.; Yehya, N. A.; and Levy, K. Y. 2024. Dynamic Byzantine-Robust Learning: Adapting to Switching Byzantine Workers. In *Forty-first International Conference on Machine Learning*. PMLR.

El-Mhamdi, E. M.; Farhadkhani, S.; Guerraoui, R.; Guirguis, A.; Hoang, L.-N.; and Rouault, S. 2021. Collaborative learning in the jungle (decentralized, byzantine, heterogeneous, asynchronous and nonconvex learning). *Advances in neural information processing systems*, 34: 25044–25057.

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

Ghadimi, S.; and Lan, G. 2013. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4): 2341–2368.

Ghadimi, S.; Lan, G.; and Zhang, H. 2016. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1): 267–305.

Gorbunov, E.; Borzunov, A.; Diskin, M.; and Ryabinin, M. 2022. Secure distributed training at scale. In *International Conference on Machine Learning*, 7679–7739. PMLR.

Gorbunov, E.; Hanzely, F.; and Richtárik, P. 2021. Local sgd: Unified theory and new efficient methods. In *International Conference on Artificial Intelligence and Statistics*, 3556–3564. PMLR.

Gorbunov, E.; Horváth, S.; Richtárik, P.; and Gidel, G. 2023. Variance Reduction is an Antidote to Byzantine Workers: Better Rates, Weaker Assumptions and Communication Compression as a Cherry on the Top. In *11th International Conference on Learning Representations, ICLR 2023*.

Guo, H.; Wang, H.; Song, T.; Hua, Y.; Lv, Z.; Jin, X.; Xue, Z.; Ma, R.; and Guan, H. 2021. Siren: Byzantine-robust federated learning via proactive alarming. In *Proceedings of the ACM Symposium on Cloud Computing*, 47–60.

Guo, H.; Wang, H.; Song, T.; Ma, Y. H. R.; Jin, X.; Xue, Z.; and Guan, H. 2024. SIREN+: Robust Federated Learning with Proactive Alarming and Differential Privacy. *IEEE Transactions on Dependable and Secure Computing*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Huang, W.; Shi, Z.; Ye, M.; Li, H.; and Du, B. 2024. Self-Driven Entropy Aggregation for Byzantine-Robust Heterogeneous Federated Learning. In *Forty-first International Conference on Machine Learning*.

Johnson, R.; and Zhang, T. 2013. Accelerating Stochastic Gradient Descent using Predictive Variance Reduction. In Burges, C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2021. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2): 1–210.

Karimireddy, S. P.; He, L.; and Jaggi, M. 2021a. Byzantine-Robust Learning on Heterogeneous Datasets via Bucketing. In *International Conference on Learning Representations*.

Karimireddy, S. P.; He, L.; and Jaggi, M. 2021b. Learning from history for byzantine robust optimization. In *International Conference on Machine Learning*, 5311–5319. PMLR.

Karimireddy, S. P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; and Suresh, A. T. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, 5132–5143. PMLR.

Khaled, A.; Mishchenko, K.; and Richtárik, P. 2020. Tighter theory for local SGD on identical and heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, 4519–4529. PMLR.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Konečný, J.; McMahan, H. B.; Ramage, D.; and Richtárik, P. 2016. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Li, Q.; Diao, Y.; Chen, Q.; and He, B. 2022. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th international conference on data engineering (ICDE)*, 965–978. IEEE.

Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3): 50–60.

Li, Z.; Lin, T.; Shang, X.; and Wu, C. 2023. Revisiting weighted aggregation in federated learning with neural networks. In *International Conference on Machine Learning*, 19767–19788. PMLR.

Malinovsky, G.; Gorbunov, E.; Horváth, S.; and Richtárik, P. 2023. Byzantine robustness and partial participation can be achieved simultaneously: Just clip gradient differences. In *Privacy Regulation and Protection in Machine Learning*.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. PMLR.

Mhamdi, E. M. E.; Guerraoui, R.; and Rouault, S. 2018. The hidden vulnerability of distributed learning in byzantium. *arXiv preprint arXiv:1802.07927*.

Mishchenko, K.; Islamov, R.; Gorbunov, E.; and Horváth, S. 2023. Partially personalized federated learning: Breaking the curse of data heterogeneity. *arXiv preprint arXiv:2305.18285*.

Moulines, E.; and Bach, F. 2011. Non-Asymptotic Analysis of Stochastic Approximation Algorithms for Machine Learning. In Shawe-Taylor, J.; Zemel, R.; Bartlett, P.; Pereira, F.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

Nguyen, T. D.; Rieger, P.; De Viti, R.; Chen, H.; Brandenburg, B. B.; Yalame, H.; Möllering, H.; Fereidooni, H.; Marchal, S.; Miettinen, M.; et al. 2022. {FLAME}: Taming backdoors in federated learning. In *31st USENIX Security Symposium (USENIX Security 22)*, 1415–1432.

Park, J.; Han, D.-J.; Choi, M.; and Moon, J. 2021. Sageflow: Robust federated learning against both stragglers and adversaries. *Advances in neural information processing systems*, 34: 840–851.

Qin, T.; and Liu, T. 2013. Introducing LETOR 4.0 Datasets. *CoRR*, abs/1306.2597.

Recht, B.; and Ré, C. 2013. Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation*, 5(2): 201–226.

Reddi, S.; Charles, Z.; Zaheer, M.; Garrett, Z.; Rush, K.; Konečnỳ, J.; Kumar, S.; and McMahan, H. B. 2020. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*.

Robbins, H.; and Monro, S. 1951. A stochastic approximation method. *The annals of mathematical statistics*, 400–407.

Rodríguez-Barroso, N.; Martínez-Cámara, E.; Luzón, M. V.; and Herrera, F. 2022. Dynamic defense against byzantine poisoning attacks in federated learning. *Future Generation Computer Systems*, 133: 1–9.

Sadiev, A.; Borodich, E.; Beznosikov, A.; Dvinskikh, D.; Chezhegov, S.; Tappenden, R.; Takáč, M.; and Gasnikov, A. 2022. Decentralized personalized federated learning: Lower bounds and optimal algorithm for all personalization modes. *EURO Journal on Computational Optimization*, 10: 100041.

Shalev-Shwartz, S.; and Ben-David, S. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Smith, V.; Chiang, C.-K.; Sanjabi, M.; and Talwalkar, A. S. 2017. Federated multi-task learning. *Advances in neural information processing systems*, 30.

Stich, S. U. 2019. Unified optimal analysis of the (stochastic) gradient method. *arXiv preprint arXiv:1907.04232*.

Tieleman, T.; and Hinton, G. 2012. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*, 6.

Tupitsa, N.; Horváth, S.; Takáč, M.; and Gorbunov, E. 2024. Federated Learning Can Find Friends That Are Beneficial. *arXiv preprint arXiv:2402.05050*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Verbraeken, J.; Wolting, M.; Katzy, J.; Kloppenburg, J.; Verbelen, T.; and Rellermeyer, J. S. 2020. A survey on distributed machine learning. *Acm computing surveys (csur)*, 53(2): 1–33.

Wagner, P.; Strodthoff, N.; Bousseljot, R.-D.; Kreiseler, D.; Lunze, F. I.; Samek, W.; and Schaeffter, T. 2020. PTB-XL, a large publicly available electrocardiography dataset. *Scientific data*, 7(1): 1–15.

Wang, S.; and Ji, M. 2022. A unified analysis of federated learning with arbitrary client participation. *Advances in Neural Information Processing Systems*, 35: 19124–19137.

Wang, Y.; Sun, T.; Li, S.; Yuan, X.; Ni, W.; Hossain, E.; and Poor, H. V. 2023. Adversarial attacks and defenses in machine learning-empowered communication systems and networks: A contemporary survey. *IEEE Communications Surveys & Tutorials*.

Woodworth, B. E.; Patel, K. K.; and Srebro, N. 2020. Minibatch vs local sgd for heterogeneous distributed learning. *Advances in Neural Information Processing Systems*, 33: 6281–6292.

Wu, Z.; Ling, Q.; Chen, T.; and Giannakis, G. B. 2020. Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks. *IEEE Transactions on Signal Processing*, 68: 4583–4596.

Xie, C.; Koyejo, O.; and Gupta, I. 2020. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In *Uncertainty in Artificial Intelligence*, 261–270. PMLR.

Xie, C.; Koyejo, S.; and Gupta, I. 2019. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *International Conference on Machine Learning*, 6893–6901. PMLR.

Xu, X.; and Lyu, L. 2020. A reputation mechanism is all you need: Collaborative fairness and adversarial robustness in federated learning. *arXiv preprint arXiv:2011.10464*.

Yan, H.; Zhang, W.; Chen, Q.; Li, X.; Sun, W.; Li, H.; and Lin, X. 2024. Recess vaccine for federated learning: Proactive defense against model poisoning attacks. *Advances in Neural Information Processing Systems*, 36.

Yang, H.; Fang, M.; and Liu, J. 2021. Achieving linear speedup with partial worker participation in non-iid federated learning. *arXiv preprint arXiv:2101.11203*.

Yin, D.; Chen, Y.; Kannan, R.; and Bartlett, P. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, 5650–5659. Pmlr.

Zhang, Z.; Cao, X.; Jia, J.; and Gong, N. Z. 2022. Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2545–2555.