An Experimental Study of Byzantine-Robust Aggregation Schemes in Federated Learning

SUBMISSION DATE / POSTED DATE

09-04-2022 / 11-04-2022

DOI

10.36227/techrxiv.19560325.v1

An Experimental Study of Byzantine-Robust Aggregation Schemes in Federated Learning

Shenghui Li, *Student Member, IEEE,* Edith C.-H. Ngai, *Senior Member, IEEE,* and Thiemo Voigt, *Member, IEEE*

Abstract—Byzantine-robust federated learning aims at mitigating Byzantine failures during the federated training process, where malicious participants (known as Byzantine clients) may upload arbitrary local updates to the central server in order to degrade the performance of the global model. In recent years, several robust aggregation schemes have been proposed to defend against malicious updates from Byzantine clients and improve the robustness of federated learning. These solutions were claimed to be Byzantine-robust, under certain assumptions. Other than that, new attack strategies are emerging, striving to circumvent the defense schemes. However, there is a lack of systematical comparison and empirical study thereof. In this paper, we conduct an experimental study of Byzantine-robust aggregation schemes under different attacks using two popular algorithms in federated learning, *FedSGD* and *FedAvg*. We first survey existing Byzantine attack strategies, as well as Byzantine-robust aggregation schemes that aim to defend against the Byzantine attacks. We also propose a new scheme, `ClippedClustering`, to enhance the robustness of a clustering-based scheme by automatically clipping the updates. Then we provide an experimental evaluation of eight aggregation schemes in the scenario of five different Byzantine attacks. Our experimental results show that these aggregation schemes sustain relatively high accuracy in some cases, but they are not effective in all cases. In particular, our proposed `ClippedClustering` successfully defends against most attacks under independent and identically distributed (IID) local datasets. However, when the local datasets are Non-IID, the performance of all the aggregation schemes significantly decreases. With Non-IID data, some of these aggregation schemes fail even in the complete absence of Byzantine clients. Based on our experimental study, we conclude that the robustness of all the aggregation schemes is limited, highlighting the need for new defense strategies, in particular for Non-IID datasets.

Index Terms—Distributed learning, federated learning, neural networks, robustness

1 Introduction

Federated learning (FL) [1], [2], [3] is a machine learning paradigm for distributed model training on decentralized data across a set of client devices (e.g., desktops, mobile phones, IoT devices). Specifically, FL repeatedly performs the following steps: the server broadcasts the current global model to client devices; the clients then perform one or several local steps of stochastic gradient descent (SGD) using private training sets, and send the updates back to the server; then the server generates a new global model by aggregating the local updates to enable the next round of training. This paradigm allows the user devices (a.k.a. clients) to perform most of the computation, without requiring any of the participants to reveal their private training data to a centralized entity or each other. In addition, benefiting from multiple steps of local updates before uploading local updates, FL improves the communication-efficiency compared to the traditional distributed learning [4].

Despite the achievements of FL in terms of data privacy and communication-efficiency, it also opens up the parameter updating process to manipulation by the clients, which brings serious security threats to model training [5]. An important class of security threats in this context are known as Byzantine failures [5], where some of the participants are not rigorously following the protocol, but upload arbitrary parameters to the central server, for example, due to faulty communication [6], or even worse, adversaries, where malicious attackers modify the update vectors to their desire and upload them to the server [7]. We use the term "Byzantine attack" to refer to the attacks where malicious attackers upload arbitrary updates to the server in order to degrade the overall performance of the global model in FL. In typical FL algorithms (e.g., *FedAvg*) [1], the server aggregates the uploaded updates by calculating their sample mean and adds the result to the global model. However, it is well-known that the result of such aggregation scheme can be arbitrarily skewed even by a single Byzantine client [8]. The server thus requires Byzantine-robust solutions to defend against malicious clients.

In recent years, a number of Byzantine-robust techniques have been proposed [9]. They can be classified into three categories: redundancy-based schemes that assign each client redundant updates and use this redundancy to eliminate the effect of Byzantine failures [10], [11], [12], [13]; trust-based schemes that assume some of the clients or datasets are trusted for filtering and re-weighting the local model updates [14], [15], [16]; robust aggregation schemes that estimate the updates according to some robust aggregation algorithms [8], [17], [18], [19], [20], [21]. For the first category, redundancy-based schemes, in the worst case, require each node to compute $\Omega(M)$ times more updates, where M is the number of Byzantine clients [10]. This overhead is prohibitive in settings with large numbers of Byzantine clients. For the second category, the trusted clients/datasets are not always available to the server due to the concern of user data privacy.

Robust aggregation schemes, in contrast, aggregate the updates efficiently, without requiring trusted clients or datasets. However, typical schemes including `GeoMed` [18], `Krum` [17],

Shenghui Li is with the Department of Information Technology, Uppsala University, Uppsala, Sweden (e-mail: shenghui.li@it.uu.se).
Edith C.-H. Ngai (corresponding author) is with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong, China (e-mail: chngai@eee.hku.hk).
Thiemo Voigt is with the Department of Information Technology, Uppsala University, Uppsala, Sweden, and also with RISE, the Research Institutes of Sweden, Stockholm, Sweden (e-mail: thiemo.voigt@it.uu.se).

TrimmedMean [8], Median [8] and CC [20], often come with limited guarantees of Byzantine robustness (e.g., only establishing convergence to a limit, or only guaranteeing that the output of the aggregation scheme has a positive inner product with the true gradient [17], [22]) and often require other strong assumptions, such as bounded absolute skewness [8]. More importantly, recent studies reveal the vulnerability of some schemes to new attacks. For instance, the A Little Is Enough (ALIE) attack can circumvent TrimmedMean and Krum by taking advantage of empirical variance between the updates of clients if such variance is high enough [23]. The Inner Product Manipulation (IPM) attack poses a significant threat to Median and Krum by manipulating the inner product between the true gradient and the robust aggregated gradients to be negative [24]. Other schemes, such as AutoGM [19] and Clustering [21], were proposed with only empirical evaluations.

These existing aggregation schemes are evaluated using different datasets, attack types and hyper-parameters. There is a lack of empirical studies that compare different schemes of utilizing the same settings. Furthermore, the impact of data heterogeneity to robustness schemes is rarely evaluated as those schemes usually assume that all clients' local data are independent and identically distributed (IID). Therefore, there is a clear need for a comparative experimental study that offers in-depth insight into the performance of the existing Byzantine-robust schemes for FL.

To meet this need, we conduct an experimental study on the Byzantine attack and defense problem in FL based on two well-known algorithms, *FedSGD* and *FedAvg* [1], [23]. We first survey existing attack strategies and robust aggregation schemes in the literature. We further propose a new aggregation scheme ClippedClustering to address the weakness of an existing clustering-based scheme. Then we design experiments to evaluate the robustness of eight representative Byzantine-robust aggregation rules by applying five state-of-the-art attacking strategies. Our experimental results show that those aggregation rules sustain relatively high accuracy in some cases. However, they are not effective in all cases. Moreover, when the local datasets are not independent and identically distributed (Non-IID), the capability of all the aggregation rules decreases significantly. With Non-IID data, some of these aggregation rules fail even in the complete absence of Byzantine clients. Furthermore, our proposed scheme performs the best in most attack scenarios when the datasets are IID. From the evaluation, we conclude that existing aggregation rules are insufficient to meet the need of Byzantine robustness, highlighting the demand for new defense strategies in FL, especially with training on Non-IID datasets.

Our key contributions can be summarized as follows:

- We survey existing Byzantine attack strategies to compromise FL, as well as Byzantine-robust aggregation schemes that aim to defend against the Byzantine attacks.
- Based on an existing clustering-based aggregation scheme, we further propose an enhanced scheme called ClippedClustering, by applying an automatical clipping technique to mitigate the effect of amplified local updates.
- We evaluate eight robust aggregation schemes (including the proposed ClippedClustering) under five representative Byzantine attack strategies. Our experimental results show that the aggregation schemes sustain high accuracy in some cases, but have limited success in other cases, especially in the presence of Non-IID data.

The rest of this paper is organized as follows: Section 2 first formulates the problem of FL and introduces two optimization algorithms. Then Section 3 introduces the threat models evaluated in this paper. Subsequently, representative robust aggregation schemes are presented in Section 4. Section 5 presents the experiments for robust aggregation schemes, from which some notable findings are uncovered. Finally, we review related work in Section 6 and make some conclusions in Section 7.

2 Federated Learning

In this section, we first formulate the optimization problem of FL. Then we introduce two popular algorithms for solving the FL problem, one is the classic distributed SGD optimization algorithm *FedSGD* and the other is the famous communication-efficient algorithm *FedAvg*.

2.1 Problem Formulation

In FL, multiple clients collaboratively learn a shared global model using their private datasets in a distributed way, assisted by the coordination of a central server. The goal is to find a parameter vector $\boldsymbol{w}$ that minimizes the following distributed optimization model:

$$\min_{\boldsymbol{w}} F(\boldsymbol{w}) = \min_{\boldsymbol{w}} \frac{1}{K} \sum_{k \in [K]} F_k(\boldsymbol{w}), \tag{1}$$

where K is the total number of clients, the local objective $F_k(\cdot)$ can be defined as empirical risk over local data, i.e., $F_k(w) = \frac{1}{n_k} \sum_{j \in [n_k]} \ell(\boldsymbol{w}; x_{k,j})$, where $\ell(\cdot;\cdot)$ is a user-specified loss function, $x_{k,j}$ is a training sample and n_k is the size of training dataset owned by client k.

A common assumption in FL is that local training datasets can be unbalanced, i.e., clients can have different numbers of training samples [1]. However, in this paper, we assume that data are balanced, i.e., $n_1 = n_2 = \cdots = n_K$ to align with most studies that specifically focus on Byzantine robustness [8], [18], [20]. We note that one can get rid of this assumption using the re-scaling trick proposed by Li et al. [25].

2.2 Optimizations of Federated Learning

We adopt the two most popular algorithms in Byzantine robust optimization literature to solve Problem (1), i.e., *FedSGD* and *FedAvg*.

2.2.1 *FedSGD*

Stochastic gradient descent (SGD) can be applied naively to the federated optimization problem (1) [1]. As summarized in Algorithm 1 with option I, each client calculates a single mini-batch gradient and uploads it to the server in parallel at each round of training. The server then aggregates the received gradients and updates the model parameters according to the aggregated gradients. Benefiting from mini-batch of stochastic gradient calculation, this approach is computationally efficient, but it still requires a very large number of communication rounds to produce good models [1], [26]. In this paper, we refer to this algorithm as *FedSGD*, which is also known as *sync-SGD* in some related work [23], [24].

Algorithm 1 Optimization of Federated Learning

Input: $K, T, \eta_t, \boldsymbol{w}^0$
1: **for** each global round $t \in [T]$ **do**
2: **for** each client $k \in [K]$ **in parallel do**
3: $\boldsymbol{w}_k^t \leftarrow \boldsymbol{w}^t$
4: **Option I** (*FedSGD*):
5: Sample mini-batch ξ from local dataset
6: $\boldsymbol{\Delta}_k^t \leftarrow \triangledown F_k(\boldsymbol{w}_k^t, \xi)$
7: **Option II** (*FedAvg*):
8: **for** E_l local rounds, **do**
9: Sample mini-batch ξ from local dataset
10: $\boldsymbol{w}_k^t \leftarrow \boldsymbol{w}_k^t - \eta_t \triangledown F_k(\boldsymbol{w}_k^t, \xi)$
11: **end for**
12: $\boldsymbol{\Delta}_k^t \leftarrow \boldsymbol{w}_k^t - \boldsymbol{w}^t$
13: Sends Δ_k^t back to the server
14: **end for**
15: $\Delta^{t+1} \leftarrow AGG(\{\Delta_k^t\}_{k\in[K]})$
16: **Option I** (*FedSGD*):
17: $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - \eta_t \Delta^{t+1}$
18: **Option II** (*FedAvg*):
19: $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t + \Delta^{t+1}$
20: **end for**
21: **return** $\boldsymbol{w}^T$

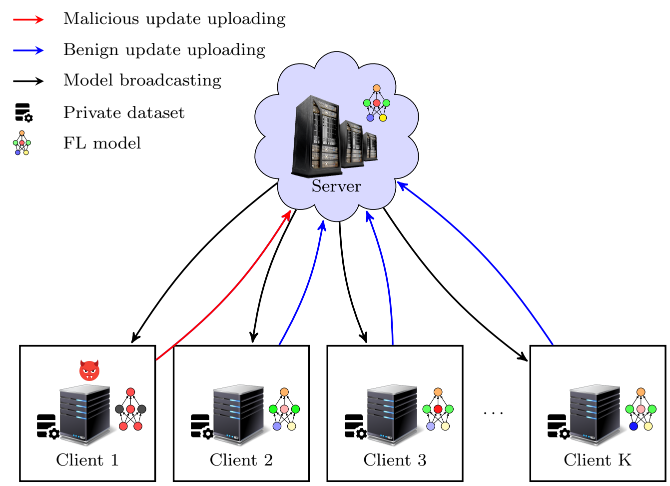


Fig. 1: Illustration of a FL system with Byzantine clients, where K clients collaboratively train a machine learning model using their local datasets. The coordinate server aggregates local updates and updates the model, then broadcasts the new model to the clients in each communication round. The attackers control some of the clients (e.g., Client 1) and send malicious updates to the server, while the honest clients compute and upload benign updates.

2.2.2 FedAvg

A more communication efficient framework for FL is *FedAvg* [1]. As summarized in Algorithm 1 with Option II, at each round of training, the server broadcasts its global model to each client. In parallel, the clients run multiple steps of SGD on their own loss functions, and send the resulting model to the server. The server then updates its global model according to its aggregation rule, and broadcast the resulting global model to each client to enable the next round of training. Multiple rounds of interactions between the server and clients are required to obtain an accurate shared global model.

As one may see, general *FedAvg*-based algorithms usually randomly select a subset of clients to perform local training while the algorithm we adopt involves full participation of all clients at each round. This is because all of the aggregation schemes considered in this paper are based on an assumption that less than half of the updates for aggregation are malicious on each round. Selecting subsets at random violates this assumption with some probability, as it may select more malicious clients than benign ones by chance. Therefore, full participation is used in this paper.

2.2.3 Update aggregation

In both *FedSGD* and *FedAvg*, the server aggregates the received updates and uses the result of the aggregation to update the global model. A widely-used aggregation scheme is calculating the sample `Mean` of the uploaded updates, i.e.,

$$\Delta^{t+1} \leftarrow \frac{1}{K}\sum_{k\in[K]} \Delta_k^t. \tag{2}$$

However, `Mean` is vulnerable to malicious local updates. As the breakdown point of `Mean` is $1/K$ [27], which means that even if only one of the clients is malicious, the resulting global model can significantly deviate from the original `Mean`. In Section 4 we will cover robust aggregations that aim to defend against malicious updates.

3 Threat Models

In this section, we describe the five threat models of Byzantine attacks we evaluate in this paper.

In terms of Byzantine attacks, most existing literature for distributed learning and federated learning focuses on convergence prevention [7], [19], [21], [24]. As illustrated in Fig. 1, the attackers (known as Byzantine clients) may upload arbitrary parameters to the server in order to degrade the performance of the global model. Thus, Line 6 and 12 of Algorithm 1 are replaced by the following:

$$\boldsymbol{\Delta}_k^t \leftarrow \begin{cases} \star & \text{if } k\text{-th client is Byzantine,} \\ \triangledown F_k(\boldsymbol{w}_k^t, \xi) & \text{if } k\text{-th client is benign (for } \textit{FedSGD}\text{),} \\ \boldsymbol{w}_k^t - \boldsymbol{w}^t & \text{if } k\text{-th client is benign (for } \textit{FedAvg}\text{),} \end{cases} \tag{3}$$

where $\star$ represents arbitrary values.

In this paper, we follow the assumption that the majority of the clients are benign [19], [20], which means we have $\frac{M}{K} < 0.5$, where M is the number of Byzantine clients. We examine five typical attacks in our threat models.

3.1 Noise

A straightforward attack is to sample some random noise from a distribution (e.g., Gaussian distribution) and add it to the updates before uploading [19], [28].

3.2 A Little is Enough (ALIE)

In contrary to the random Noise attack, the attackers may modify the noise carefully to pretend being benign and fool the aggregation rules. A Little is Enough (ALIE) [23] assumes that the benign updates are expressed by a normal distribution. The attackers therefore immediately take advantage of the high empirical variance between the updates of clients and upload a noise in a range without being detected.

For each coordinate $i \in [d]$, the attackers calculate mean (μ_i) and std (δ_i) over benign updates, and set corrupted updates Δ_i to values in the range $(\mu_i - z^{max}\delta_i, \mu_i + z^{max}\delta_i)$, where z^{max} ranges from 0 to 1, and is typically obtained from the Cumulative Standard Normal Function [23].

3.3 Inner Product Manipulation (IPM)

The Inner Product Manipulation (IPM) attack [24] seeks for the negative inner product between the true mean of the updates and the output of the aggregation schemes, so that at least the loss will not descent. Assuming that the attackers know the mean of benign updates, a specific way to perform an IPM attack is

$$\Delta_1^t = \cdots = \Delta_M^t = -\frac{\epsilon}{K-M}\sum_{i=M+1}^{K}\Delta_i^t, \tag{4}$$

where we assume that the first M clients are malicious, ϵ is a positive coefficient controlling the magnitude of malicious updates. Then the `Mean` becomes

$$\frac{1}{K}\sum_{k\in[K]}\Delta_k^t = \frac{K-M(1+\epsilon)}{K(K-M)}\sum_{i=M+1}^{K}\Delta_i^t. \tag{5}$$

Note that when $\epsilon < \frac{K}{M} - 1$, IPM does not change the direction of the average over benign updates but only decreases its magnitude, because we have

$$\frac{K-M(1+\epsilon)}{K(K-M)} > 0,$$

the optimization thus can still converge using `Mean` as an aggregation scheme. However, as we will show in Section 5, such an attack can circumvent the defense of several aggregation schemes and inverse the direction of updates, which heavily damages the global model. On the contrary, when $\epsilon > \frac{K}{M} - 1$, the sign of `Mean` is reversed, indicating that the loss will increase if the model is updated using the `Mean`. In our experiment, we examine both cases by letting $\epsilon = 0.5$ and $\epsilon = 100$, respectively.

3.4 Sign Flipping (SF)

Different from IPM, the Sign Flipping (SF) attackers do not need to know the updates from other clients and simply flip the signs of the gradient [13], [20], which means that the attackers strive to maximize the loss via gradient ascent instead of gradient descent. Specifically, in *FedSGD*, the clients upload the negative gradients; in *FedAvg*, the flipping is applied at every local updating step.

3.5 Label Flipping (LF)

The aforementioned attacks assume that the attackers have full access to the training process so that they can modify the updates immediately. However, full access may be limited as the training APIs are not always open. Correspondingly, the attackers can also change the training dataset instead of the update parameters [29]. The Label Flipping (LF) attack simply flips the label of each training sample [7]. Specifically, a label l is flipped as $L - l - 1$, where L is the number of classes in the classification problem and $l = 0, 1, ..., L - 1$.

4 Aggregation Schemes for Evaluation

In this section, we survey existing robust aggregation schemes, which represent state-of-the-art methods in the literature. Then, we propose a new scheme `ClippedClustering`, which addresses the weakness of the clustering-based scheme. Other than that, we provide a taxonomy of the eight aggregation schemes evaluated in our experiments.

All aggregation schemes considered in this paper are working on each round separately. For the sake of readability, we will omit the notation of the round $(^t)$ in this section.

4.1 Krum

`Krum` [17] strives to find one of the local model updates that is closest to another $K - M - 2$ ones with respect to squared Euclidean distance, which can be expressed by:

$$Krum := \{\boldsymbol{\Delta}_i | i = \operatorname*{argmin}_{i\in[K]} \sum_{i\to j} \|\boldsymbol{\Delta}_i - \boldsymbol{\Delta}_j\|^2\},$$

where $i \to j$ is the indices of the $K - M - 2$ nearest neighbours of $\boldsymbol{\Delta}_i$ measured by squared Euclidean distance, recall that K is the number of clients in total, and M is the number of malicious clients.

Under the *FedSGD* framework, `Krum` was proven to converge with an important assumption that $c_1\sigma < \|g\|$, where c_1 is a constant factor depending on the number of malicious clients and the dimension of model parameters, σ is the maximal variance of the updates and $\|g\|$ is the expectation of updates.

4.2 GeoMed

The Geometric Median (`GeoMed`) [18], [30] scheme aims to find a vector that minimizes the sum of its Euclidean distances to all the update vectors:

$$GeoMed := \operatorname*{argmin}_{\boldsymbol{z}} \sum_{k\in[K]} \|\boldsymbol{z} - \boldsymbol{\Delta}_k\|. \tag{6}$$

Although there is no closed-form solution to the `GeoMed` problem, a $(1+\epsilon)$-approximate solution can be computed in nearly linear time [31].

Similar to `Krum`, `GeoMed` was also proven to converge under the *FedSGD* framework, with the assumption that $c_2\sigma < \|g\|$, where c_2 is a another constant factor that differs from c_1.

4.3 AutoGM

Auto-weighted Geometric Median (`AutoGM`) [19] is a generalized version of `GeoMed`. `AutoGM` aggregates the updates by solving the following problem:

$$\begin{aligned} AutoGM := &\operatorname*{argmin}_{\boldsymbol{z}} \sum_{k\in[K]} \alpha_k \|\boldsymbol{z} - \boldsymbol{\Delta}_k\| + \frac{\lambda}{2}\|\boldsymbol{\alpha}\|^2, \\ &s.t. \quad \boldsymbol{\alpha} \in \mathbb{R}_+^K, \mathbf{1}^\top \boldsymbol{\alpha} = 1, \end{aligned} \tag{7}$$

where λ is a user-specified hyper-parameter that controls the smoothness of $\boldsymbol{\alpha}$.

The key idea of optimizing `AutoGM` is to divide the problem into two parts, i.e., one subproblem for estimating the weighted `GeoMed`, and the other subproblem for weighting the importance of each point. Then, we can minimize the objective iteratively with respect to one variable each time while fixing the other one [19].

4.4 Median

`Median` [8] is defined as the coordinate-wise median of the given set of updates, i.e.,

$$med := Median(\{\boldsymbol{\Delta}_k : k \in [K]\}),$$

where the i-th coordinate $med_i = median(\{\boldsymbol{\Delta}_k^i : k \in [K]\})$, and $median$ is the usual (one-dimensional) median.

When using the *FedSGD* framework, the robustness of the `Median` scheme is based on the assumptions that the gradient of the loss function has bounded variance, and each coordinate of the gradient has coordinate-wise bounded absolute skewness [8].

4.5 TrimmedMean

The `TrimmedMean` [8] aggregation scheme computes the coordinate-wise trimmed average of the model updates, which can be expressed by:

$$trmean := TrimmedMean(\{\boldsymbol{\Delta}_k : k \in [K]\}),$$

where the i-th coordinate $trmean_i = \frac{1}{(1-2\beta)m}\sum_{x\in U_k} x$, and U_k is a subset obtained by removing the largest and smallest β fraction of its elements.

In addition to the aforementioned assumptions for `Median`, the robustness of `TrimmedMean` relies on one stronger assumption that all the moments of the derivatives of the loss function are bounded [8].

4.6 Centered Clipping (CC)

Centered Clipping (`CC`) [20] iteratively clips the updates around the center while updating the center accordingly. For $l \geq 0$, `CC` computes

$$\Delta_{l+1} \leftarrow \Delta_l + \frac{1}{K}\sum_{k\in[K]}(\Delta_i - \Delta_l)\min(1, \frac{\tau_1}{\|\Delta_i - \Delta_l\|}), \quad (8)$$

where Δ_0 is assigned with the aggregated updates in the previous round.

Karimireddy et al. [20] proofed the robustness of the `CC` scheme when the variance of updates is bounded and $\frac{K}{M} \leq 0.15$.

4.7 Clustering

This `Clustering` aggregation scheme [21], [32] first calculates the pairwise cosine similarities between their parameter updates, i.e.,

$$\alpha_{i,j} := \frac{<\boldsymbol{\Delta}_i, \boldsymbol{\Delta}_j>}{\|\boldsymbol{\Delta}_i\|\|\boldsymbol{\Delta}_j\|}, \quad (9)$$

then it separates the client population into two groups based on the cosine similarities

$$c_1, c_2 \leftarrow \underset{c_1\cup c_2=[K]}{\text{argmin}}\,\{\max_{i\in c_1\cup j\in c_2}\alpha_{i,j}\}. \quad (10)$$

Finally, it aggregates the updates in the larger groups between c_1 and c_2 using `Mean`.

Despite the lack of theoretical guarantee of robustness, `Clustering` achieves superior robustness in some cases, as we will show in Section 5. However, an obvious drawback of clustering using cosine similarities is that it only considers the relative directions, ignoring the magnitude of each vector. The attackers thus can fool the clustering scheme by simply amplifying the updates without changing their directions. As a consequence, the resulting updates added to the parameters will make the model jump over the minima and prevent the convergence of the optimization without being detected.

Defense	Euclidean distance	Mean	Median	Cosine similarity	Clipping
`Krum` [17]	✓				
`GeoMed` [18]	✓		✓		
`AutoGM` [19]	✓		✓		
`Median` [8]			✓		
`TrimmedMean` [8]		✓			
`CC` [20]		✓			✓
`Clustering` [21]		✓		✓	
`ClippedClustering` (ours)		✓		✓	✓

TABLE 1: Summary of the robust aggregation schemes and their main defense mechanisms.

4.8 ClippedClustering

We enhance the robustness of the aforementioned `Clustering` aggregation scheme by performing a clipping on all the updates before clustering, i.e.,

$$\Delta_k \leftarrow \Delta_k \min(1, \frac{\tau}{\|\Delta_k\|}). \quad (11)$$

Here, τ is a clipping value hyper-parameter that needs to be carefully chosen by the end-user. Note that this clipping scheme is the so-called clip by norm, not clip by value, where individual values of the update vectors are clipped if they go beyond a pre-set value. In clip by norm, the entire update is scaled if the norm of the update exceeds the threshold τ. Thus we place a maximum on the magnitude of each vector that can be taken during training, preventing the attackers from amplifying the updates in the same direction. If the norm of update is below the threshold τ, the update is unaffected.

Inspired by [33], we design an automatic clipping strategy to defend against potential amplified malicious updates that the naive cosine similarity-based clustering scheme cannot handle well. Specifically, we set the clipping value hyper-parameter based on the statistics of the historical norms of the updates uploaded during training, i.e., we save the update norms up to current iteration and automatically set τ using the 50-th percentile value of the history.

4.9 Taxonomy

Table 1 shows a taxonomy of the eight aggregation schemes, where `Krum`, `GeoMed`, and `AutoGM` are typical Euclidean distance-based schemes, i.e., they are all designed to find a vector closest to the updates measured by Euclidean distance. Among them, `GeoMed` and `AutoGM` are both based on geometric median. The `Median` scheme simply computes the coordinate-wise median instead of geometric median. `TrimmedMean`, `CC`, `Clustering` and `ClippedClustering` are all categorized as mean-based schemes as they eventually compute the mean, although they also utilize other mechanisms. `Clustering` and `ClippedClustering` both perform clustering based on cosine similarity while `ClippedClustering` clips the updates before clustering.

5 Evaluation

In this section, we design experiments to evaluate the aforementioned robust aggregation schemes under the five different attacks and show the experimental results.

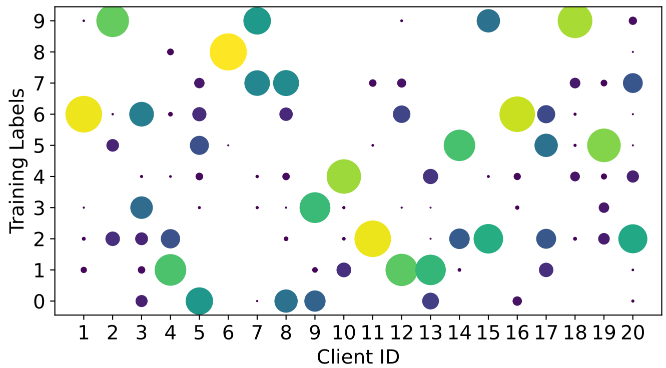


Fig. 2: Visualization of statistical heterogeneity among clients on our Non-IID partition of CIFAR-10 dataset, where the x-axis indicates client IDs, the y-axis indicates class labels, and the size of scattered points indicates the number of training samples for a label available to that client.

5.1 Experimental Setup

We simulate an FL system with the server and 20 clients, targeting at an image classification task on the CIFAR-10 dataset [1] with both IID and Non-IID partitions, where eight of the clients are Byzantine clients. For the IID partitioning, we randomly split the training set into 20 subsets and allocate them to the 20 clients, thus each client has 2500 training samples. For Non-IID partition, we follow prior work [34], [35] and model the Non-IID data distributions with a Dirichlet distribution $\boldsymbol{p}_l \sim Dir_K(\alpha)$, then we allocate a $\boldsymbol{p}_{l,k}$ proportion of the training sample of class l to client k, in which a smaller α indicates stronger Non-IID data partition. We let $\alpha = 0.1$ and visualize the resulting statistical heterogeneity of labels in Fig. 2. Such a partition is strongly Non-IID as one can see that some of the classes are completely missing for each client.

In the experiments, we choose a lightweight Compact Convolutional Transformers (CCT) network [36], as such a small yet effective model has more potential to overcome the on-board resource limitation of FL devices [37]. We train the model for 7500 and 400 communication rounds for *FedSGD* and *FedAvg*, respectively. The batch size is 32 by default. As suggested by Karimireddy et al. [20], we decay the learning rate during training to improve convergence, i.e., for *FedSGD*, we let

$$\eta_t \leftarrow \begin{cases} 0.1 & \text{if } t \leq 3750, \\ 0.05 & \text{if } 3750 < t \leq 5000, \\ 0.025 & \text{else;} \end{cases} \tag{12}$$

for *FedAvg*, we let

$$\eta_t \leftarrow \begin{cases} 0.1 & \text{if } t \leq 75, \\ 0.05 & \text{if } 75 < t \leq 100, \\ 0.025 & \text{else.} \end{cases} \tag{13}$$

For *FedAvg*, we apply 50 SGD steps before uploading the updates to the server (i.e., $E_l = 50$).

5.2 Impact on the `Mean` Scheme

We first demonstrate the impact of attacks on conventional *FedSGD* and *FedAvg* using the `Mean` scheme to aggregate updates by plotting test accuracy versus the number of communication rounds in Fig. 3. At first glance, when the datasets are IID, *FedAvg* takes fewer communication rounds than *FedSGD* to converge when there is no attack, benefiting from multiple steps of local training. However, when the datasets are Non-IID, the performance of *FedAvg* significantly decreases while *FedSGD* still maintains a relatively high accuracy. We note that this is a well-known drawback of *FedAvg* [38], [39].

Recall that some of the attacks (e.g., IPM with small ϵ [24]) are particularly designed to break the line of robust defense, which means that they may not bring much damage to the `Mean` scheme. For instance, Fig. 3 shows that IPM with $\epsilon = 0.5$ produces less damage to `Mean` compared to the other attacks, as the malicious updates do not change the direction of the average update but only decrease its magnitude (see Section 3.3). Furthermore, Noise and IPM ($\epsilon = 100$) eventually damage the models under the four settings by decreasing the test accuracy to around 10% (no better than random guess). This is because they both make large changes to the updates, and `Mean` could be easily biased by large changes [19].

5.3 Impact on Robust Aggregation Schemes

Fig. 4 depicts the overall comparison of the robust aggregation schemes in *FedSGD* and *FedAvg* with respect to test accuracy on the IID partitioned CIFAR-10 dataset.

First of all, Euclidean-based schemes (i.e., `Krum`, `GeoMed`, and `AutoGM`) reach slightly lower accuracy compared to the other schemes in the complete absence of attackers. This might be because that they all tend to select a single update that is closest to all or part of the others measured by Euclidean distance. Furthermore, they all show similar robustness in both *FedSGD* and *FedAvg*, e.g., with IID data partition, they all handle Noise and IPM ($\epsilon = 100$) well while somehow struggling with the other attacks. This is not surprising as those Euclidean-based schemes are essentially designed to defend against large changes of updates. On the other hand, small scale attacks such as IPM with $\epsilon = 0.5$ challenges them as the malicious updates are close to the benign ones when measured by Euclidean distance.

In this experiment, cosine similarity-based schemes, i.e., `Clustering` and our proposed `ClippedClustering`, successfully defend against more types of attacks than the other schemes. For instance, in *FedAvg*, `Clustering` results in around 80% test accuracy under all attacks except SF. `ClippedClustering` further mitigates this attack and successfully defends against all types of attacks.

Other schemes show, to certain extent, robustness in some cases while they fail in others. For example, `Median` sustains relatively high accuracy in *FedAvg* under the Sign Flipping (SF) attack compared to all the other schemes except `ClippedClustering`. `TrimmedMean` achieves similar performance as `Median`, except that it shows vulnerability to the SF attack in *FedAvg*.

Another important observation is that ALIE successfully circumvents all aggregation schemes of *FedSGD* in our experiments and keeps the test accuracy below 20%, as shown in Fig. 4a. However, according to Fig. 4b, the models suffer much less damage from ALIE when trained with *FedAvg*. Moreover, even the `Mean` scheme could handle it well (Fig. 3). We thus infer that multiple steps of SGD updates may result in lower variance compared to single-step updates when the datasets are IID.

Fig. 5 further shows the comparison based on Non-IID data partition. It can be seen that the accuracy significantly drops for all

1. https://www.cs.toronto.edu/ kriz/cifar.html

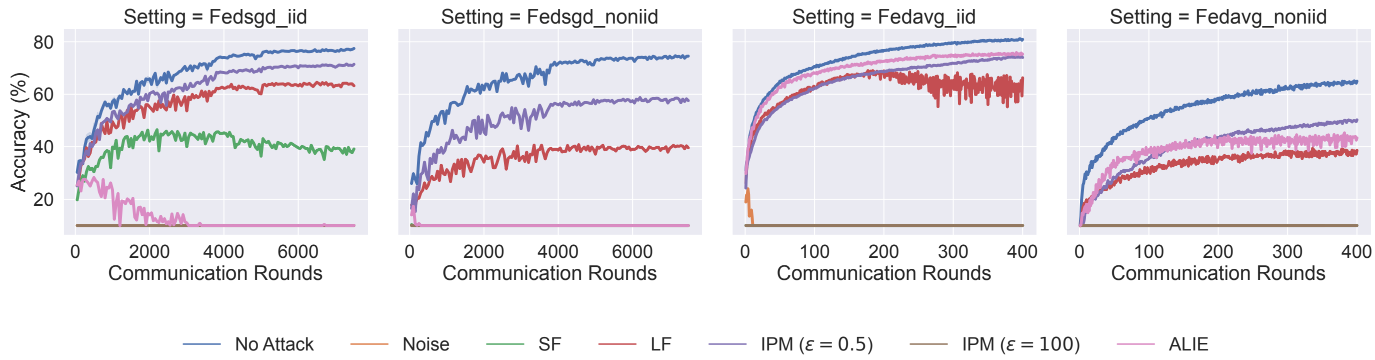
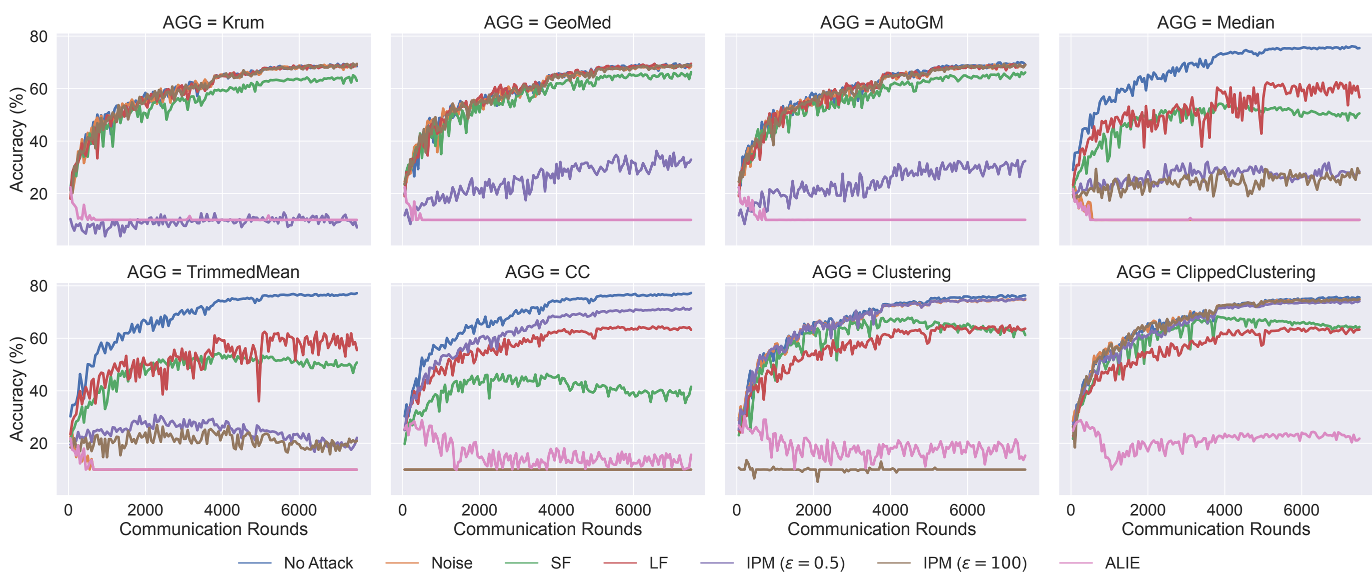
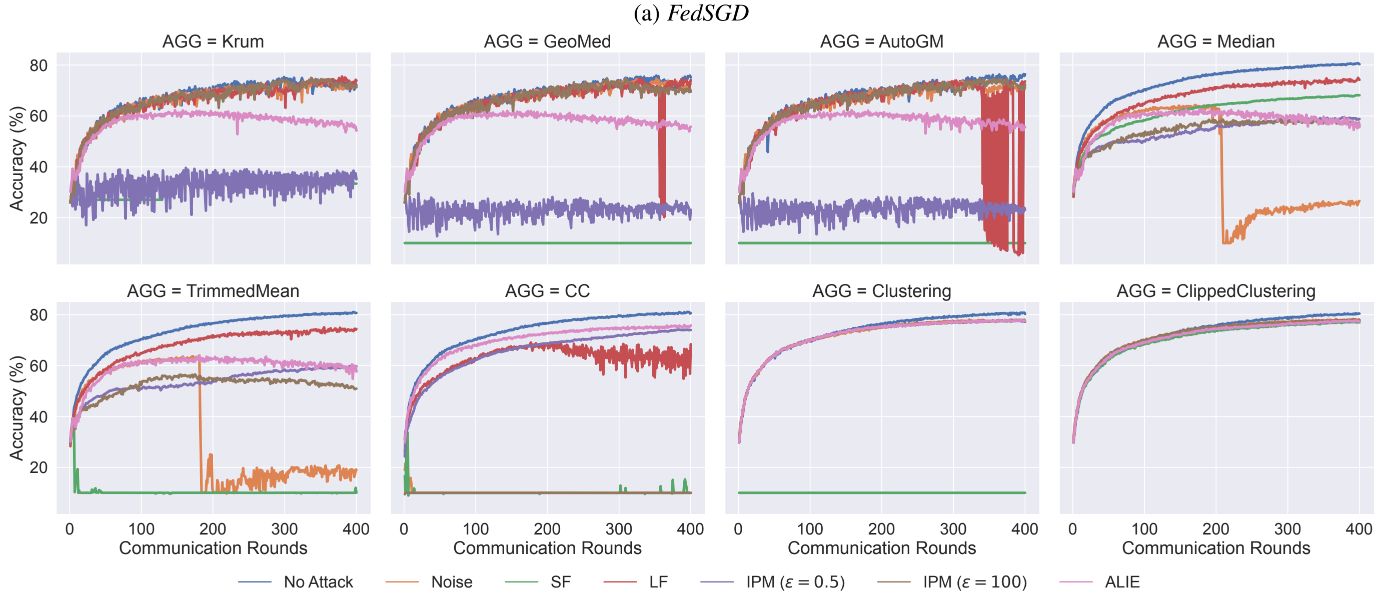


Fig. 3: Impact of malicious attacks on the `Mean` aggregation scheme. In this experiment, training with Non-IID datasets is more vulnerable to attacks, compared with using IID datasets.



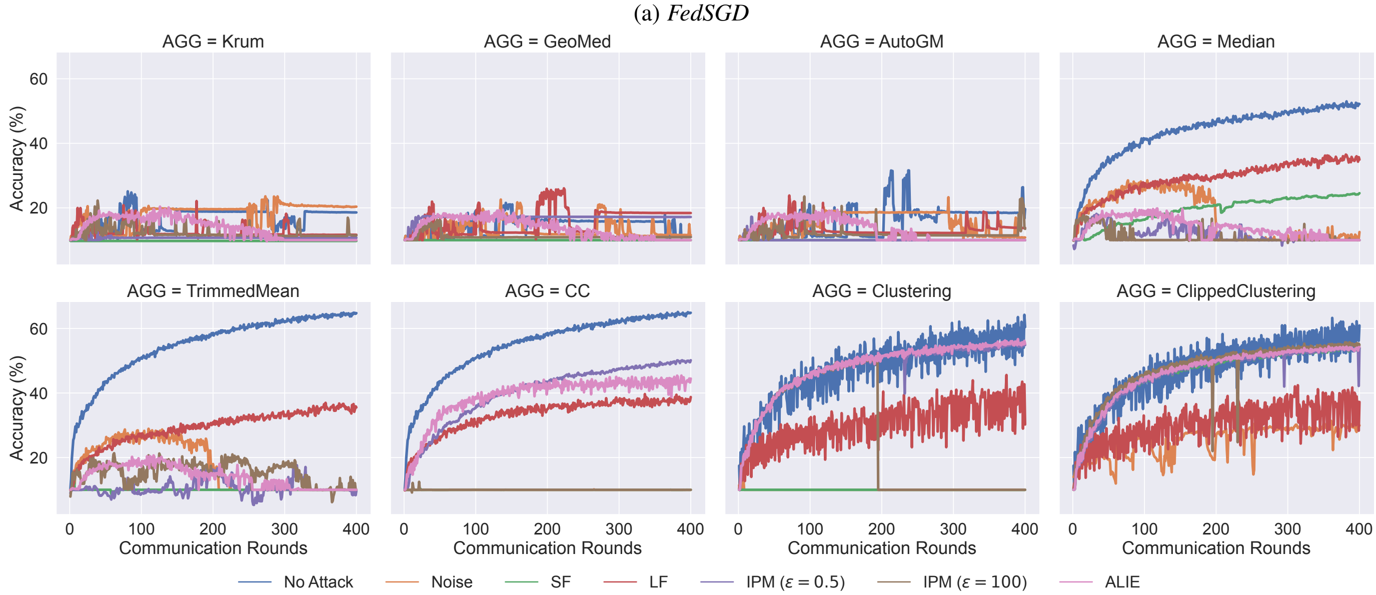
(a) *FedSGD*



(b) *FedAvg*

Fig. 4: Performance on IID datasets. These schemes sustain high accuracy only in some of the cases. Particularly, `ClippedClustering` successfully defends against all examined attacks in *FedAvg* while still suffering from ALIE in *FedSGD*.

(a) *FedSGD*



(b) *FedAvg*

Fig. 5: Performance on Non-IID datasets. None of the aggregation schemes could cope well with all the attacks. In addition, `Krum`, `GeoMed` and `AutoGM` fail to perform well even in the complete absence of Byzantine clients.

aggregation schemes compared with Fig. 4. Specifically, Euclidean-based schemes (i.e., `Krum`, `GeoMed` and `AutoGM`) fail to achieve satisfying performance even in the complete absence of Byzantine clients in both *FedSGD* and *FedAvg*. Other schemes also lose their capability to guard the model under certain attacks.

5.4 Impact of Batch Size

Note that in the previous experiments, the batch size per client is relatively small (i.e., 32), which leads to a large variance among benign updates, thus making the attacks more challenging [20]. Now we investigate the effect of batch size on the robustness of aggregation schemes by ranging batch size in $\{32, 128, 512, 2500\}$. Fig. 6 shows the performance of *FedSGD* with IID partition under ALIE attacks. `Mean`, `Clustering`, `ClippedClustering`, and `CC` become more robust as the batch size increases. This is because the variance tends to be lower as we increase the batch size. Particularly, when the batch size is 2500, the clients use all their training data for each step of training, stochastic gradient then becomes population gradient (the optimization becomes full-batch gradient descent). However, all the other aggregation schemes still fail to achieve acceptable test accuracy. Although a large batch size is more favorable for robustness purposes as indicated by this experiment, it is not the desired solution as it brings a large computation burden to local training.

5.5 Impact of the Number of Malicious Clients

To study the impact of the number of malicious clients, we perform an experiment using *FedAvg* with IID data under SF attacks (which dramatically challenges the performance as already shown in Fig. 4b). As shown in Fig. 7, `GeoMed`, `AutoGM`, `TrimmedMean`, and `Clustering` surprisingly fail in the presence of even only two malicious clients, most of which perform well in *FedSGD* with

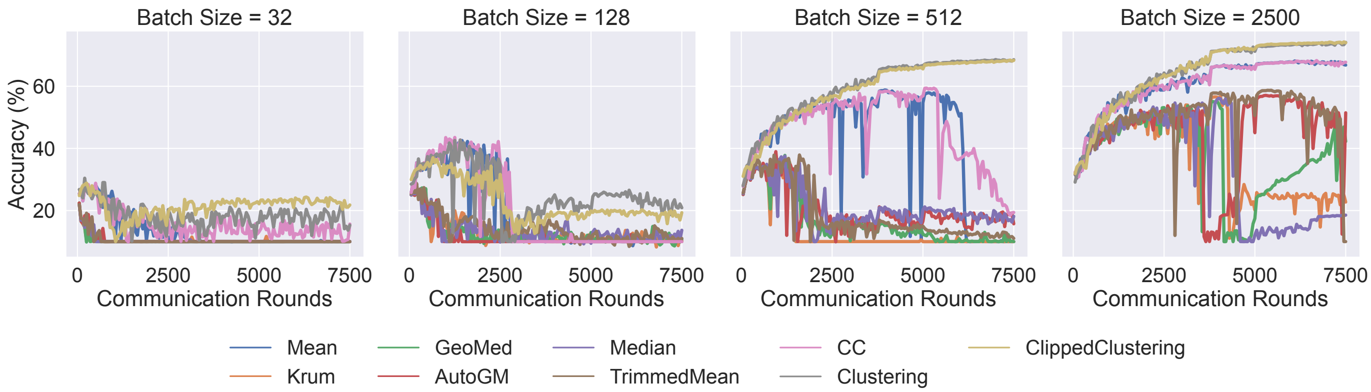


Fig. 6: Impact of batch size on the performance of *FedSGD* with IID partition under ALIE attack. The aggregation schemes (including `Mean`) become more robust as the batch size increases.

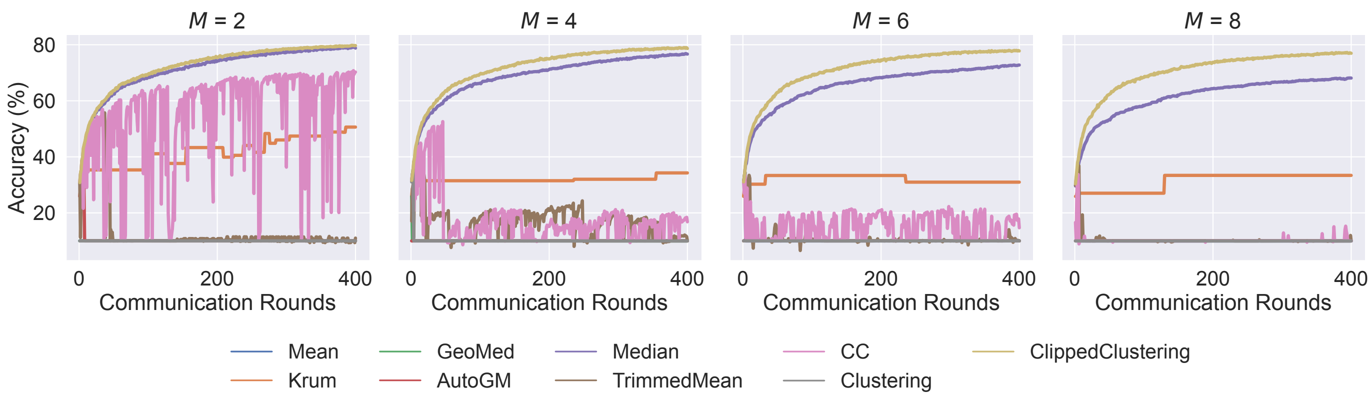


Fig. 7: Impact of number of malicious clients on the performance of *FedAvg* with IID partition under SF attack. Only `Median` and `ClippedClustering` consistently show robustness.

up to eight malicious clients, according to Fig. 4a. Nevertheless, the proposed `ClippedClustering` scheme consistently performs the best with various numbers of malicious clients.

5.6 Pairwise Cosine Similarities

Recall that the aggregation schemes show different robustness in *FedSGD* and *FedAvg* with respective to IID and Non-IID data partitions. To further investigate this issue, we compare the pairwise cosine similarities of all benign local updates without attacks. We note that cosine similarly reflects the angle between two vectors, i.e., higher value indicates smaller angle. As visualized in Fig. 8, the pairwise cosine similarities of updates from *FedSGD* vary widely, no matter if the local datasets are IID or not. A considerable pairs of clients even show negative similarities. Such a stochastic property may confuse the robust aggregation schemes and make it more challenging to detect malicious updates. The updates from *FedAvg* with IID data show the highest pairwise similarities, which means that their update directions are almost identical. Benefiting from this, clustering-based schemes can group benign updates together and exclude malicious updates. However, they become less similar when local datasets are Non-IID.

6 Related Work

We divide relate work into two parts: Byzantine attacks on FL and Byzantine-robust FL.

6.1 Byzantine Attacks on FL

Byzantine attacks on FL are carried out by malicious clients during the distributed optimization of machine learning models, aiming to bias the global model to the desire of those malicious clients [5]. Depending on the adversarial goals, Byzantine attacks in FL can be classified into two categories: targeted attacks and untargeted attacks [5], [29]. Targeted attacks, such as backdoor attacks, aim to make the global model generate attacker-desired misclassifications for some particular test samples [40], [41], [42]. While untargeted attacks aim to degrade the overall performance of the global model indiscriminately [7].

We particularly focus on untargeted attacks as most Byzantine-robust studies do [8], [17], [18], [19], [20], [21]. Many studies considered to launch attacks by adding Gaussian noise or flipping the sign of the actual updates [13], [19]. Those attacks, however, can be detected by some Euclidean distance-based aggregation schemes such as `Krum` [17], as they usually make the malicious updates far from the benign ones when measured by Euclidean distance. On the other hand, Baruch et al. [23] showed that the attackers can actually circumvent robust schemes including `TrimmedMean` and `Krum` by taking advantage of empirical variance between the updates of clients if such variance is high enough. Furthermore, Xie et al. proposed an attack strategy, Inner Product Manipulation (IPM) attack that poses a significant threat to `Median` and `Krum` by manipulating the inner product between the true mean of the updates and the output of the aggregation schemes [24].

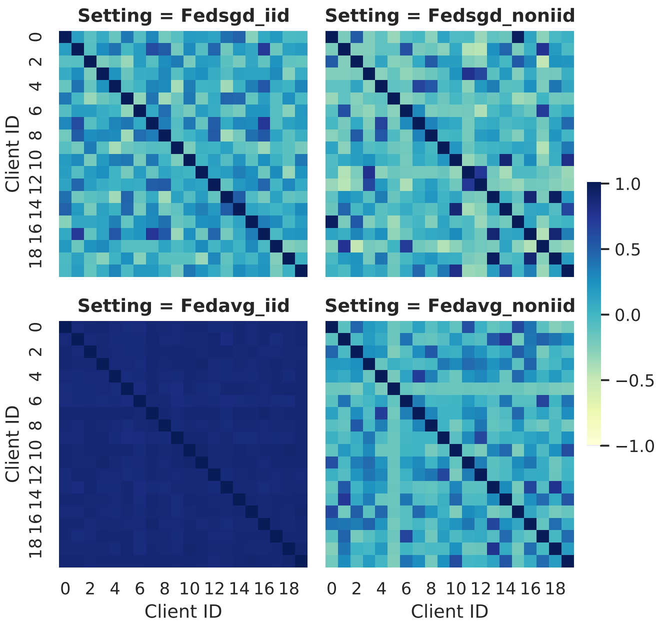


Fig. 8: Pairwise cosine similarities between local updates in different scenarios, where a similarly value 1 represents identical update directions, and -1 indicates opposite directions. *FedAvg* with IID data results in the most similar directions.

6.2 Byzantine-robust FL

In FL settings, a number of strategies have been explored to defend against specific types of attacks or failures, including backdoor attacks [42], [43], free-rider attacks [44], [45], and gradient inversion attacks [46], [47]. On a more general level, Byzantine-robust FL solutions aim to mitigate the effect of arbitrary updates uploaded by malicious clients, instead of focusing on specific types of attacks [7]. Those Byzantine-robust solutions can be classified into three categories: redundancy-based schemes, trust-based schemes, and robust aggregation schemes.

Redundancy-based schemes assign each client redundant updates and use this redundancy to eliminate the effect of Byzantine failures [10], [11], [48]In 2018, Chen et al. [10] presented a framework, DRACO, for robust distributed training that uses ideas from coding theory. In DRACO, each client evaluates redundant gradients that are used by the server to eliminate the effects of adversarial updates. In 2019, Rajput et al. presented DETOX, a framework that combines algorithmic redundancy with robust aggregation. The defense of DETOX operates in two steps, a filtering step that uses limited redundancy to significantly reduce the effect of Byzantine nodes, and a hierarchical aggregation step that can be used in tandem with any state-of-the-art robust aggregation method. However, these redundant updates, in the worst case, require each node to compute $\Omega(M)$ times more updates, where M is the number of Byzantine clients [10]. This overhead is prohibitive in settings with a large numbers of Byzantine clients.

Trust-based schemes assume that some of the clients or datasets are trusted for filtering and re-weighting the local model updates [13], [14], [15], [16]. For example, in 2019, Li et al. [13] proposed to incorporate the objective function with a regularization term, which minimizes the distance between the server parameters and the client parameters. In 2021, Park et al. [15] designed an entropy-based filtering scheme to detect the outlier updates based on some trusted public data at the server-side. During the training, the server computes the entropy of each update with the trusted dataset. Based on their experimental observations, they argue that the updates with higher entropy will lead to lower accuracy during the testing stage. Thus, they set a threshold for the entropy and filter out updates with entropy higher than the threshold. In 2021, Cao et al. [16] utilized cosine similarity to measure the similarity between updates submitted by the clients and the update obtained by training based on the trusted dataset owned by the server. The authors argued that an attacker can manipulate the directions of updates to perform model poisoning attacks, and the directions of the updates can, to a certain extent, indicate the honesty of the end devices. After the calculation of cosine similarity, the server calculates a trust score for each update using the ReLu function. The score is then used as the weight for the global model aggregation. In general, trust-based schemes have the potential to deal with the situation where more than half of the updates are malicious according to some pre-validated information to detect malicious updates. However, trusted datasets or clients are not always available for the server, for example, due to the concern of user data privacy.

Robust aggregation schemes estimate the global update based on the local updates according to their robust aggregation rules or algorithms [8], [17], [18], [19], [20], [21]. Byzantine-robust aggregation has been explored to handle the devices sending corrupted updates to the server, including geometric median (`GeoMed`) [18], `Krum` [17], `TrimmedMean` [8], and `Median` [8]. They are commonly used to estimate the model parameters and mitigate the effect of malicious updates in global aggregation. In 2017, Chen et al. [18] proposed a `GeoMed`-based method to aggregate the gradients for distributed statistical machine learning and showed the robustness and convergence in i.i.d settings. In 2020, Wu et al. [49] showed that `GeoMed` scheme provably provides improved Byzantine robustness compared to other aggregation schemes in FL. In 2022, Pillutla et al. [30] applied `GeoMed` as a robust aggregation rule for FL and analyze the convergence of the resulting FL algorithm for least-squares objective with IID local datasets. In 2022, Li et al. [19] proposed `AutoGM`, a variant of `GeoMed` that automatically re-scales the weight of each parameter component according to a user-specified threshold of skewness. According to our empirical study in this paper, this category of schemes all show limitations in *FedSGD* and *FedAvg* algorithms in terms of Byzantine robustness.

7 Conclusions

In this paper, we provided an experimental study of Byzantine robust aggregation schemes for FL. In particular, we survey existing Byzantine attacks and defense strategies in the FL literature. We also propose a novel scheme, `ClippedClustering`, which enhances the robustness of clustering-based scheme by automatically clipping the updates to mitigate the effect of amplified malicious updates. We then evaluate eight robust aggregation schemes under five representative Byzantine attack strategies. Our experimental results show that all those aggregation schemes achieve limited robustness in the presence of Byzantine attacks. In the future, it would be interesting to improve the robustness of FL from more perspectives, e.g., low variance algorithms, and robust learning rates.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized

data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[3] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[4] Y. Arjevani and O. Shamir, "Communication complexity of distributed convex learning and optimization," *Advances in neural information processing systems*, vol. 28, 2015.

[5] L. Lyu, H. Yu, J. Zhao, and Q. Yang, "Threats to federated learning," in *Federated Learning*. Springer, 2020, pp. 3–16.

[6] F. Ang, L. Chen, N. Zhao, Y. Chen, W. Wang, and F. R. Yu, "Robust federated learning with noisy communication," *IEEE Transactions on Communications*, vol. 68, no. 6, pp. 3452–3464, 2020.

[7] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[8] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018.

[9] S. Hu, J. Lu, W. Wan, and L. Y. Zhang, "Challenges and approaches for mitigating byzantine attacks in federated learning," *arXiv preprint arXiv:2112.14468*, 2021.

[10] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos, "Draco: Byzantine-resilient distributed training via redundant gradients," in *International Conference on Machine Learning*. PMLR, 2018, pp. 903–912.

[11] S. Rajput, H. Wang, Z. Charles, and D. Papailiopoulos, "Detox: A redundancy-based framework for faster and more robust gradient aggregation," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[12] J.-y. Sohn, D.-J. Han, B. Choi, and J. Moon, "Election coding for distributed learning: Protecting signsgd against byzantine attacks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14 615–14 625, 2020.

[13] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1544–1551.

[14] N. Konstantinov and C. Lampert, "Robust learning from untrusted sources," in *International conference on machine learning*. PMLR, 2019, pp. 3488–3498.

[15] J. W. Park, D.-J. Han, M. Choi, and J. Moon, "Sageflow: Robust federated learning against both stragglers and adversaries," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[16] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *ISOC Network and Distributed System Security Symposium (NDSS)*, 2021.

[17] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[18] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2017.

[19] S. Li, E. Ngai, and T. Voigt, "Byzantine-robust aggregation in federated learning empowered industrial iot," *IEEE Transactions on Industrial Informatics*, 2021.

[20] S. P. Karimireddy, L. He, and M. Jaggi, "Learning from history for byzantine robust optimization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5311–5319.

[21] F. Sattler, K.-R. Müller, T. Wiegand, and W. Samek, "On the byzantine robustness of clustered federated learning," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8861–8865.

[22] R. Guerraoui, S. Rouault *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3521–3530.

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

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

[25] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[26] Z. Li, V. Sharma, and S. P. Mohanty, "Preserving data privacy via federated learning: Challenges and solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 3, pp. 8–16, 2020.

[27] H. P. Lopuhaa and P. J. Rousseeuw, "Breakdown points of affine equivariant estimators of multivariate location and covariance matrices," *The Annals of Statistics*, pp. 229–248, 1991.

[28] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6357–6368.

[29] M. S. Jere, T. Farnan, and F. Koushanfar, "A taxonomy of attacks on federated learning," *IEEE Security & Privacy*, vol. 19, no. 2, pp. 20–28, 2020.

[30] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Transactions on Signal Processing*, 2022.

[31] M. B. Cohen, Y. T. Lee, G. Miller, J. Pachocki, and A. Sidford, "Geometric median in nearly linear time," in *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, 2016, pp. 9–21.

[32] Z. Li, L. Liu, J. Zhang, and J. Liu, "Byzantine-robust federated learning through spatial-temporal analysis of local model updates," *arXiv preprint arXiv:2107.01477*, 2021.

[33] P. Seetharaman, G. Wichern, B. Pardo, and J. Le Roux, "Autoclip: Adaptive gradient clipping for source separation networks," in *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2020, pp. 1–6.

[34] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2351–2363, 2020.

[35] S. Li, E. Ngai, F. Ye, and T. Voigt, "Auto-weighted robust federated learning with corrupted data sources," *ACM Trans. Intell. Syst. Technol.*, feb 2022, just Accepted. [Online]. Available: https://doi.org/10.1145/3517821

[36] A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li, and H. Shi, "Escaping the big data paradigm with compact transformers," *arXiv preprint arXiv:2104.05704*, 2021.

[37] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[38] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5132–5143.

[39] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," *arXiv preprint arXiv:2003.13461*, 2020.

[40] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *International Conference on Learning Representations*, 2019.

[41] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948.

[42] S. Andreina, G. A. Marson, H. Möllering, and G. Karame, "Baffle: Backdoor detection via feedback-based federated learning," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 852–863.

[43] C. Zhao, Y. Wen, S. Li, F. Liu, and D. Meng, "Federatedreverse: A detection and defense method against backdoor attacks in federated learning," in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, 2021, pp. 51–62.

[44] J. Lin, M. Du, and J. Liu, "Free-riders in federated learning: Attacks and defenses," *arXiv preprint arXiv:1911.12560*, 2019.

[45] Y. Fraboni, R. Vidal, and M. Lorenzi, "Free-rider attacks on model aggregation in federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 1846–1854.

[46] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, "Evaluating gradient inversion attacks and defenses in federated learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[47] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients–how easy is it to break privacy in federated learning?" *arXiv preprint arXiv:2003.14053*, 2020.

[48] D. Data, L. Song, and S. Diggavi, "Data encoding for byzantine-resilient distributed gradient descent," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2018, pp. 863–870.

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

Shenghui Li received the B.S. degree from Xidian University, Xi'an, China, in 2017, and the M.S. degree from Sun Yat-sen University, Guangzhou, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of Information Technology, Uppsala University, Uppsala, Sweden.

His current research interests include federated learning, distributed optimization, and statistical machine learning.

Edith C.-H. Ngai is currently an Associate Professor in Department of Electrical and Electronic Engineering, The University of Hong Kong. Before joining HKU in 2020, she was an Associate Professor in the Department of Information Technology, Uppsala University, Sweden. Dr. Ngai was a guest researcher at Ericsson Research Sweden in 2015-2017. Previously, she has conducted research in Imperial College London, Simon Fraser University, Tsinghua University, and UCLA. Her research interests include Internet-of-Things, machine learning, data analytics, and smart cities. Dr. Ngai is a VINNMER Fellow (2009) awarded by Swedish Governmental Research Funding Agency VINNOVA. She led the "Green IoT" project in Sweden, which was named on the IVA's 100-list from the Royal Swedish Academy of Engineering Sciences in 2020. She is currently an Area Editor of IEEE Internet of Things Journal and an Associate Editor of IEEE Access and IEEE Transactions of Industrial Informatics.

Thiemo Voigt received the Ph.D. degree from Uppsala University, Sweden, in 2002. He is currently a Professor of computer science with the Department of Information Technology, Uppsala University. He also leads the Networked Embedded Systems Group, RISE Computer Science. His current research interests include low-power networking, system software for embedded networked devices and the Internet of Things. His work has been cited more than 17900 times. He is a member of the editorial board for the IEEE IoT Newsletter and ACM Transactions on Sensor Networks (TOSN).