Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Defending against model poisoning attack in federated learning: A variance-minimization approach

Hairuo Xu, Tao Shu [*]

Department of Computer Science and Software Engineering, Auburn University, AL, 36849, United States

ARTICLE INFO

Keywords:
Federated learning
Model poisoning attack defense
Non-I.I.D.
Variance-minimization
Robust aggregation

ABSTRACT

The distributed nature of federated learning (FL) renders the learning process susceptible to model poisoning attacks, whereby local workers in FL report fabricated and false local training outcomes to the FL server with the intention to compromise/degrade the global model, or even derail the global learning process so that it can no longer converge. Existing defense mechanisms typically consider the falsified local updates as outliers that reside far away from the mean update, and attempt to counter against such model poisoning attacks by detecting and eliminating those outliers in the reported local training updates. These methods do not perform well when the data of different workers are non-I.I.D. and/or when there are multiple colluding attackers, under which an outlier update is not always a falsified update, and vice versa. In this paper, we propose a novel defense mechanism, MinVar, to counter the model poisoning attacks in FL from a drastically different perspective. Instead of detecting and eliminating outlier local updates from the global model aggregation, MinVar takes all local updates but assigns different weights to them in the global model aggregation. MinVar decides the optimal weights by formulating and solving an optimization problem in each iteration of the learning process, which aims to suppress the contribution of those falsified (i.e., malicious) updates while still retaining the contribution of those honest/truthful updates. Based on the sparsity observation in most deep neural networks, a data sampling technique is further proposed to reduce the computation complexity of MinVar while preserving its defense performance. Extensive experiments are conducted on both the MNIST and CIFAR-10 datasets, and the results verify the effectiveness of the proposed MinVar defense model.

1. Introduction

Federated Learning (FL) is known to be vulnerable to model poisoning attacks. This is due to the distributed nature of the FL: the convergence and global training outcome of FL is determined by the local training outcome of every individual worker that participates in the FL in each iteration. Because workers are only sharing their local training outcomes (model parameters or gradients) in FL, the truthfulness of each worker's local data, and consequently the truthfulness and correctness of the local training outcomes that are being reported to the global model aggregation process, is hard to be guaranteed. Therefore, defense methods that are able to filter out the suspicious local updates to protect the overall FL are highly desirable.

Most existing model-poisoning defense mechanisms such as [1–5] rely on the assumption that a falsified local training update is an outlier that resides far away from the mean update (the distance in this case can be defined as, e.g., the cosine similarity between the updates). As a result, these defense mechanisms try to counter against the model poisoning attacks by detecting and eliminating those outliers in the reported local training updates from the global model aggregation, as illustrated in Fig. 1(a). These methods perform well when the data at each worker are I.I.D., but will have limited performance when the data of different workers are non-I.I.D., whereby an outlier update is not always a falsified update, and vice versa. Eliminating an outlying but truthful update in this case is a mistake, which will degrade the generalization of the global model.

Similar issue also arises when there are multiple colluding attackers that coordinate their training outcome reports to manipulate the outlier-detection process. For example, a set of coordinated false updates could well bias the centroid of the whole population of updates. As a result, a truthful update that is not an outlier in the attack-free case may now become an outlier and hence be (mistakenly) eliminated from the aggregation. Similarly, a false update that could have been identified as an outlier in the absence of colluding attackers may now be considered as an inlier, and hence is (mistakenly) admitted to participate in the aggregation. It is clear that existing defense methods will fail in either case.

* Corresponding author.
E-mail address: tshu@auburn.edu (T. Shu).

https://doi.org/10.1016/j.jisa.2024.103744

Available online 11 March 2024

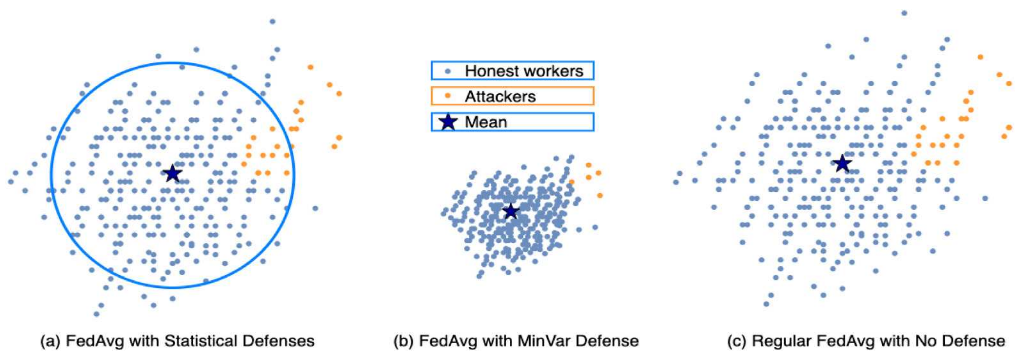

Fig. 1. The local updates utilized for FedAvg on the global server in three scenarios. (a) FedAvg with statistical (existing) defense methods that draw a boundary (blue circle) based on mean and standard deviation and kick out all the outside updates, (b) FedAvg with MinVar defense that minimize the variance between all local updates, and assign small weights to suspicious workers (attackers), and (c) Regular FedAvg with No defense. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Keeping the above weaknesses of the outlier-elimination-based methods in mind, in this paper we propose a novel defense mechanism, MinVar, to combat model poisoning attacks in FL from a drastically different perspective. Instead of detecting and eliminating outlier local updates from the model aggregation, MinVar takes all local updates but assigns different weights to them in the model aggregation. MinVar decides the optimal weights by formulating and solving an optimization problem in each iteration with the objective of minimizing the variance of the weighted local updates. The optimization formulation takes as input the local updates from all workers, and calculates the optimal aggregation weights for these workers. The weighted local updates are then aggregated to update the global model.

As will be clear shortly, the weight assignment by MinVar is distance related. Intuitively, under MinVar, an outlying local update is still considered being suspicious, and hence would be assigned a smaller weight (as shown in Section 5). This will allow this outlying local update to participate in the model aggregation, but at a diluted/discounted contribution. Such a "soft" filtering method well compensates the weaknesses of the "hard" filtering (i.e., admit/reject) adopted in existing defense mechanisms. Specifically, in the non-I.I.D.-data scenario, an honest worker that has a very different local dataset will still be able to contribute positively to the global model. On the other hand, in the colluding-attacker case, as long as the attackers are still minorities (i.e., the number of attackers is relatively small when compared with that of the honest workers), the false updates from attackers that are meant to bias the centroid of the whole population of updates will increase the variance of the population. The weight assignment by MinVar will tend to counter/suppress the influence of these false updates, since the objective function pursued by MinVar is to minimize the variance of the updates after they are weighted. The above main idea of MinVar is illustrated in Fig. 1(b), where it can be seen that with the optimized aggregation weights, the weighted local updates are condensed while the original layout (shown in 1(c)) is still preserved.

The main contribution of this work includes the following three folds:

- A novel defense mechanism, MinVar, is proposed to counter against model-poisoning attacks in FL. MinVar does not require the calculation and detection of outliers, and therefore is applicable to both I.I.D. and non-I.I.D. dataset cases. To the best of our knowledge, MinVar is the first defense mechanism in the literature that considers/counters the model poisoning attacks in FL in the non-I.I.D. case.
- We analyze the time complexity of MinVar and propose a sampling technique to reduce its complexity based on the sparsity observation in most (deep) neural networks. We show that the proposed MinVar algorithm that runs on the sub-samples achieves comparable performance to that run on the full local updates, while the computation time required to solve the optimization problem formulated by MinVar is proportional to the sample rate (i.e., percentage sampled).
- We evaluate the performance of MinVar and the proposed sampling technique by comparing with existing defense methods, via extensive FL experiments over various datasets. The results show that our proposed algorithm achieves significantly better defense effects than its counterparts, especially for the cases that there are multiple colluding attackers and/or the local data of workers is non-I.I.D.

The rest of the paper is organized as follows. The existing attack and defense methods for federated learning are reviewed in Section 2. We describe the model definition, more specifically, the backbone federated learning framework used, and the attack models targeted in Section 3. We explain the details of the proposed MinVar method, and analyze the time complexity of original and the sub-sampled version of MinVar in Section 4. The performance evaluation of MinVar, in comparison with other existing methods are shown in Section 5, and finally we conclude our paper in Section 6.

2. Related works

2.1. Poisoning attacks in federated learning

Federated learning is known to be vulnerable due to its nature of crowd-sourcing the datasets by local workers and aggregation by the server. In general, the poisoning attacks in FL can be classified into two categories, the data poisoning attack, where an adversary focuses on manipulating the dataset to infect the gradients to be uploaded to the server, and model poisoning attack, where the adversary manipulates the uploaded model (parameters and gradients) to infect the training of the FL.

The data poisoning attacks could be further divided into two groups: the dirty label attacks and the clean label attacks. Dirty label attacks involves the change of the label, and consists of the label flipping attacks and backdoor attacks. [6,7] are the representative works of the label flipping attack where the attacker changes the labels of a subset of the training data, causing the model to learn from mislabeled examples. Another type of the dirty label attacks is the backdoor attack [8,9], where the attacker insert a trigger as a feature into existing data samples, and then set the label in correspondence with the trigger, leading to the wrong prediction of the attacker-selected label once such trigger is presented in the input. The clean label attacks can be further divided into the clean label backdoor attack (with trigger), and the trigger-less attack. The former is similar to the dirty label backdoor attack except that the attacker cannot modify the label of the data samples [10–12]. The trigger-less attack [13–15], on the other hand, disrupts the feature space of the targeted class by injecting well-crafted poisoned data samples.

In model poisoning attacks, in contrast to the data poisoning attack, an adversary can directly manipulate the gradients to be uploaded in order to inject poisoned neurons into the global model. The model poisoning attacks could be further divided into three categories based on the attackers goals: the backdoor attack, the model update attack, and the privacy attack. In backdoor attacks, the adversary injects the predefined backdoor trigger (and/or Trojan) into the model, causing the model to make incorrect predictions when the trigger is present. Such poisoning parameters (or gradients) could be achieved by weight manipulation [16], bit-wise flipping [17], and solving a local optimization problem [18–20]. The model update attack exploits the vulnerabilities in the model update or the fine-tuning process, where the attacker inject the malicious parameters/gradients into the model, leading to the performance degradation of partial or entire model. The model update attack could be further divided into two groups based on the way they obtain the poisoning parameters. [21,22] belongs to the first group, in which the attacker formulates and solves a local optimization problem where the solution is the poisoning parameters. The second group trains a separate local helper model based on the information downloaded from the server (which contains other worker's training outcome), and generate the poisoned parameters based on other workers' training outcome [23,24]. In the privacy attack, the adversary manipulates the parameters to gain private information about other participants. Such privacy attack includes the membership inference attack [25,26] which goal is to determine whether certain data samples exist in the target worker's dataset, and label distribution inference attack which tries to infer other workers' training labels [27,28].

2.2. *Defenses (robust aggregation) mechanisms in federated learning*

With the development of the aforementioned attack models, several defense (or robust aggregation) methods were proposed to counter those attack models. Krum and Multi-Krum [1] selects the workers gradients to be aggregated based on the intuition that poisoned workers' gradients should reside far away from the honest ones. Trimmed Mean [2] aggregates each dimension of the input gradients separately by sorting each dimension first, and then removing the β largest and smallest values and only aggregate the rest. Unliked the intuition from Krum and Multi-Krum, Bulyan [3] shows that the gradients from an adversary may be close to the benign ones, but obtain a very large value for a certain dimension, and thus prevent the overall convergence of the entire federated learning. Based on such observation, the authors propose Bulyan defense, which requires a lot more number of honest peers to enforce enough "goodness" in the aggregated parameters. Median [4] is another statistical defense method which aggregates the median of each dimension. Rather than considering the similarities of magnitude like the above methods, Adaptive Federated average (AFA) [29] takes the multi-dimensional directions into consideration by comparing the Cosine Similarity and discards the ones that are out of bound.

We consider the above defense models as statistics-based "hard" filtering defense methods as they all compute a decision boundary and completely eliminate those outside the boundary. We argue that such defense methods won't work well when the data distribution for each individual workers is non-I.I.D., since it could eliminate a worker who has seemingly irregular, but true data samples that reflect its temporal and spatial information.

Several recent defense methods [30–33] also considered the non-I.I.D. scenarios in FL. Mini-FL [30] proposes a novel framework that consider the non-I.I.D. case by first partitioning the gradients into groups based on a grouping feature such as IP address, timestamp and user ID (which are also regarded as the causes of the non-I.I.D. characteristic of the FL training data). After that, since the gradients within the same group are considered statistically similar, the existing I.I.D. defense methods (such as Krum, Median, …,etc.) could be applied within each group to eliminate the outliers (potential attackers). Finally, the remaining gradients are weighted based on the group population and are aggregated on the server. Sageflow [31] and FLTrust [33] both require the server to collect a clean validation dataset used as the "root of trust" (or reference) for the local model updates, and then based on the distance from the reference, each local update is assigned a corresponding weight (or score) used for the global aggregation. However, the effectiveness of both algorithms depend on whether the data distribution of the validation dataset resembles the FL training dataset. If the validation dataset collected by the server shares similar distribution as the FL training dataset, then both algorithm can successfully defend against the attackers. Such a requirement is hard to meet in real applications as the data distribution of the entire training dataset is most likely unknown due to the distributed nature of FL. ShieldFL [32] proposes the defense in FL from the privacy-preserving perspective. The server decides the weight of each encrypted local model updates by calculating the cosine similarity score, and perform the weighted aggregation on the server.

In this paper, we present MinVar, a "soft" filtering defense model considering the non-I.I.D. scenario that takes all local gradients, but with different weights, where the weights specify how trustworthy each worker is. Unlike the existing defense methods that consider the non-I.I.D. case, our algorithm does not require an additional step of gradients grouping, or the prior knowledge of the overall data distribution. Our algorithm also address the scenario where there are multiple attackers (i.e., the Byzantine attack), and show that we achieve higher defense capacity (i.e., we are able to defend against more attackers) than the existing defense methods.

3. Model definition

3.1. *Federated learning*

There are several variations of FL, and in this paper, we consider the most general scenario, the parameter server model [34]. The FL system contains one server, which is responsible for maintaining the global parameters denoted as $\mathbf{w}_{\text{global}}$, and n collaborators (also referred to as workers or peers in the following text) with worker IDs $\{0, 1, 2, \ldots, n-1\}$. There exists a communication protocol that allows the collaborators to share the training information through local gradients upload (to the server) and global parameters download (from the server). All workers had agreed in advance on the machine learning task to collaborate and to use the same local training architecture. In this paper, we assume the training architecture is (deep) neural networks for its outstanding performance in many machine learning tasks. Each peer maintains its own private dataset, which will not be shared with other peers due to privacy and security concerns.

The federated learning process starts by the initialization at iteration (or epoch) 0, where the global server and the local workers correspondingly set their model parameters $\mathbf{w}^0_{\text{global}} \in \mathbb{R}^d$, and $\mathbf{w}^0_i \in \mathbb{R}^d$, where i denotes the worker ID. After that, the training is performed iteratively. More specifically, at iteration t, each worker starts training by first downloading and replacing all (or partial) local parameters with the global parameters from the server. Such download and replace procedure can be expressed as $\mathbf{w}^t_i \leftarrow \mathbf{w}^{t-1}_{\text{global}}$, where $\mathbf{w}^{t-1}_{\text{global}}$ is the global parameters maintained on the server at the end of iteration $t-1$. Then, each worker performs local training, and uploads the full local training gradients (also referred to as local updates in the following text), denoted as $\mathbf{g}^t_i \in \mathbb{R}^d$, to the server. The server collects the local updates from all workers in epoch t, and performs the FedAvg algorithm to update the global parameters as:

$$\mathbf{w}^t_{\text{global}} = \mathbf{w}^{t-1}_{\text{global}} - \alpha \frac{1}{n} \sum_{i}^{n} \mathbf{g}^t_i \tag{1}$$

where α denotes the learning rate. And thus, one iteration is said to be finished.

3.2. Attack model

We consider an attack model similar to the one proposed in [35]. The attackers are assumed to have full access to the compromised workers, i.e., the attacker is able to modify the worker's dataset and the local training outcome reported over the communication link with the server. In each iteration, the attacker(s) upload fabricated report(s) (also referred as attack vectors in following texts), inline with the distribution of the true training outcome in order to elude from detection. Assume there are m colluded attackers, whose worker ID $i \in \{CompromisedWorkers\}$. In each iteration t, the attackers start by regular local training like the honest workers, and obtain the true local training gradients $\mathbf{g}_i^t$. After that, the attackers' local gradients are shared within the colluded group, and the mean and standard deviation of the shared local gradients are calculated, denoted as $\boldsymbol{\mu}^t$, and $\boldsymbol{\sigma}^t$. Then, each element of the attack vector $\mathbf{g}_a^t$, denoted as g_{aj}^t, where $0 \le j \le d$ denotes the dimension ID, is computed by $g_{aj}^t = \mu_j^t + z\sigma_j^t$, where μ_j^t and σ_j^t are the jth elements of $\boldsymbol{\mu}^t$ and $\boldsymbol{\sigma}^t$, respectively, and z can be regarded as the scale of the attack. Such attack vector is distributed to each attacker to be uploaded to the server, pretended as the true local updates, and aims to derail the global convergence of the federated learning system. The authors in [35] draw such z from the Cumulative Standard Normal Distribution, and use the maximum z found. In contrast with their method, we employ a range of scales (z values) that covers/includes their z^{max}. Such a treatment allows us to evaluate the effectiveness of the proposed MinVar defense mechanism under different attack strengths, including those that have attack scales greater than z^{max}.

It is obvious that the above method requires the existence of at least two attackers in order to calculate the mean and standard deviation vectors $\boldsymbol{\mu}^t$ and $\boldsymbol{\sigma}^t$. For the special case that there is only a single attacker, we tweak the above attack mechanism as follows: Instead of computing the mean and standard deviation vectors over the attackers (i.e., over m), the mean and standard deviation of the attacker, which are two scalars, say μ^t and σ^t, are computed over the elements of the attacker's local gradients (there are d of them). The attack vector is then created by generating a d-dimension random vector whose d elements are drawn respectively from a Gaussian distribution of mean μ^t and standard deviation $z\sigma^t$. Such a method is able to generate an attack vector that conforms with the distribution of the true gradients of the attacker, and thus preserves the essence of the attack strategy for the more general multi-attacker case.

4. MinVar defense method

4.1. General design of MinVar

We argue that the regular FedAvg algorithm with equally assigned weights to each worker, i.e., $\frac{1}{n}$, in Eq. (1) is not capable of effectively capturing or learning the temporal and spatial information contained in the local updates, especially when the data at each source are non-I.I.D. Furthermore, when there are attackers, simply averaging the local updates from the honest workers and the suspicious vectors uploaded by adversaries could lead to the failure of convergence in many cases.

Inspired by the fact mentioned previously that large variance makes federated learning vulnerable, and meanwhile embracing the ineffective-equal-weight in FedAvg, our defense method aims to evaluate the uploaded information (could be true local gradients from honest peers or attack vectors from adversaries) from each worker, and assign each of them an aggregation weight based on the evaluation results. Regarding each of the uploaded vector as a data sample, we train an online regression model with the objective that minimizes the variance between all uploaded information. Denote

- g_{ij}^t to be the element at the jth dimension of local update ($\mathbf{g}_i^t$),
- $\boldsymbol{\mu}_j^t$ to be the mean of all local updates ($\mathbf{g}_i^t$) at the jth dimension over all the workers, and
- h_i^t to be the weight assigned to each local update ($\mathbf{g}_i^t$) from worker i.

Our objective function is to find the best $\mathbf{h}$ at each iteration t that

$$\underset{\mathbf{h}}{\text{minimize}} \quad \sum_j^d \sum_i^n (h_i \cdot g_{ij} - \mu_j)^2, \text{where } \mu_j = \frac{\sum_i^n (h_i \cdot g_{ij})}{n} \tag{2}$$

We can simplify Eq. (2) as

$$\underset{\mathbf{h}}{\text{minimize}} \quad \sum_j^d \sum_i^n (h_i^2 \cdot g_{ij}^2 + \frac{(\sum_i^n h_i \cdot g_{ij})^2}{n^2} - 2h_i \cdot g_{ij} \cdot \mu_j). \tag{3}$$

Considering the real application of the federated learning scenario, an equality and an inequality constraints are added to Eq. (3), where we enforce all the weights to be greater than 0, and sum to 1. We also add an $L2$ normalization term to avoid the potential weight domination problem, i.e., say 1 is assigned to one worker and 0s are assigned to the rest. With the constraints and the normalization term added, we can formulate the following MinVar optimization problem:

$$\begin{aligned} \underset{\mathbf{h}}{\text{minimize}} \quad & \sum_j^d \sum_i^n (h_i^2 \cdot g_{ij}^2 + \frac{(\sum_i^n h_i \cdot g_{ij})^2}{n^2} - 2h_i \cdot g_{ij} \cdot \mu_j) + \lambda \|\mathbf{h}\|_2 \\ \text{s.t.} \quad & \sum_i^n h_i = 1, \\ & h_i \ge 0. \end{aligned} \tag{4}$$

where λ is the normalization coefficient. This problem can be solved by Quadratic Programming, and we employ the cvxnp [36] package to perform the optimization.

With the solution to Eq. (4), we can rewrite Eq. (1), the aggregation phase on the server as

$$\mathbf{w}_{\text{global}}^t = \mathbf{w}_{\text{global}}^{t-1} - \alpha \sum_i^n h_i^t \cdot \mathbf{g}_i^t \tag{5}$$

where by having h_i^t for each individual local updates at each iteration, the FL system is now able to capture the temporal and spatial information obtained in each local dataset, and defend against up to the state-of-the-art-defined maximum attackers. The proposed MinVar defense method could be summarized as Algorithm 1.

Algorithm 1 MinVar: Variance-Minimization Defense

```
1: for t=0 ... end of FL training iterations do
2:     Collect local updates: $\{\mathbf{g}_0^t, \mathbf{g}_1^t, \mathbf{g}_2^t, ..., \mathbf{g}_{n-1}^t\}$
3:     Solve the optimization problem in Eq. (4) and get $\mathbf{h}^t$ : $\{h_0^t, h_1^t, h_2^t, ..., h_{n-1}^t\}$
4:     Perform global aggregation according to Eq. (5)
5: end for
```

Our intuition in trying to minimize the variance of the weighted local updates comes from the observation that larger variance over local updates makes FL more vulnerable, as it allows larger high-dimensional space for an attacker to fabricate feasible attack vectors that reside further away from the mean than some honest workers, but closer to the mean than other honest workers. For example, in Fig. 1(c), some of the attack vectors (yellow dots) are closer to the mean than some of the honest workers (blue dots) are, especially at the bottom left areas. Such attack vectors skew the true mean to a new, incorrect position, and could lead to the failure of FL convergence [35]. Minimizing the overall variance between the weighted local updates before aggregation will shrink the feasible spaces for the above attack vectors, and therefore enhance the security of FL.

4.2. Complexity analysis and reduction

It is worth noting that the necessity to learn a regression model (or solve a quadratic problem) at each iteration of FL could introduce a

Algorithm 2 MinVar Defense with Sub-Sampling

```
Require: $r$, the percentage to be sampled
1: for t=0 ... end of FL training iterations do
2:     Uniformly random sample $r\%$ dimensions.
3:     for each $i \in \{WorkerID\}$ do
4:         $\tilde{\mathbf{g}}_i^t = \mathbf{g}_i^t$[indices of $r$]
5:     end for
6:     Collect local updates: $\{\tilde{\mathbf{g}}_0^t, \tilde{\mathbf{g}}_1^t, \tilde{\mathbf{g}}_2^t, ..., \tilde{\mathbf{g}}_{n-1}^t\}$
7:     Continue with line 3 of Algorithm 1
8: end for
```

delay, and such delay could increase enormously with the size up of the local gradients (depended on the training architecture). It has been studied that gradient-based methods can achieve the time complexity of $\mathcal{O}(k)$ to solve a general quadratic problem for each solving iteration [37, 38], where k denotes the computational cost for one iteration.

To ensure the scalability and the low computational cost of our MinVar defense mechanism, we implement the sub-sampling technique prior to the training of the regression model. In each FL iteration, instead of calculating the corresponding aggregation weights (h_i) for each worker using the entire local updates ($\mathbf{g}_i$), we randomly sample a partition of the local updates, denoted as $\tilde{\mathbf{g}}_i$ and utilize the sub-samples to compute the aggregation weights, i.e., replace g_{ij} by $\tilde{g_{ij}}$ in Eq. (4). Algorithm 2 describes the MinVar algorithm with the sub-sampling technique applied. The intuition behind such sub-sampling procedure is the fact that most large-scale (deep) neural networks are sparse and with many zero-value parameters. We show that such sub-sampling method won't affect the preciseness of the learned weights ($\mathbf{h}$) from the proposed defense mechanism in Section 5.4, in contrast to the weights learned by full gradients.

As mentioned previously, the time overhead of the original MinVar algorithm for each iteration can be approximated as $\mathcal{O}(k)$. It's worth noting that k is largely dependent on the number of samples utilized in one epoch. For example, for a quadratic problem where the Stochastic Gradient Descent method [39] obtains the complexity of $\mathcal{O}(k)$ in one iteration, vanilla gradient descent method could take $\mathcal{O}(vk)$, where v denotes the total number of samples utilized for computation in each iteration [37]. Similarly, in our case, by sub-sampling the local updates which formulates a smaller-scale quadratic problem, the delay could be reduced to $\mathcal{O}(r\% \cdot k)$ for each solving iteration, where $r\%$ denotes the percentage sampled from the full local updates. Our experiments confirms such speed up in Section 5.4.

5. Experiments

5.1. Dataset and experimental setup

We test our proposed MinVar defense method on the classification tasks with the MNIST [40] and the CIFAR10 [41] datasets. For each dataset, we model the non-I.I.D data distribution on workers using the Dirichlet distribution with beta=0.5, where not only the total number of samples, but also the number of samples in each class are differently distributed for each worker. Both the single-attacker scenario, where among n peers, there exists only one attacker ($m = 1$), and the multiple-attackers scenarios, where there exists up to $m = \lceil n/2 \rceil - 1$ attackers are considered and implemented in the experiments.

5.2. Validation for single-attacker defense

Figs. 2 and 3 demonstrates the performance of our MinVar defense method when the variance-based attack is injected on the MNIST and CIFAR10 datasets, respectively. In each figure, the subfigures (a), (b) and (c) represent different settings where $n = 10, 15$, and 20, respectively. These figures show that under all studied cases, the proposed MinVar method successfully defends against such attacks by saving the FL system under attack from not converging at all to the optimal convergence point (around 95% for MNIST and around 60% for CIFAR10).

In comparison with the existing Bulyan defense method (shown in orange bars), it can be seen that MinVar outperforms Bulyan by up to 56% in MNIST and saves the diverged FL on CIFAR10. We also found that the FL system with Bulyan defense implemented actually harms the overall accuracy when there is no attacker (i.e., when the attack scale is 0) as shown in Fig. 2 on MNIST, and same observation could be made in Fig. 3 on the CIFAR10 dataset. This can be caused by the fact that Bulyan was not designed for non-I.I.D. scenarios. It suffers from misclassifying the honest peers with different dataset as attackers, and therefore kicks them out, leading to the degradation the global accuracy. Furthermore, comparing Figs. 2 and 3, it can be seen that with the increased difficulty of dataset, the performance of Bulyan is worse, since more false alarms could be generated.

As the main idea of our MinVar defense is to learn how some of the local updates differ from others, and assign corresponding weights to each, we also show the heatmap of the weights learned in each scenarios. Notice that our MinVar is implemented at each epoch during the training, but for simplicity, only the weights learned at the last epoch are shown in Figs. 4 and 5. It is demonstrated that the worker (peer) with ID 2 was given near-zero weights when an attack truly exists (scales 50–200 on the x-axis), which corresponds to our pre-defined attacker ID.

5.3. Validation for multiple-attacker defense

The experiments for the multi-attacker scenarios are also implemented and shown in Figs. 6 and 7. The x-axis depicts the attack scales $0-10$, where 0 specifies the attack-free scenario. Similar observations can be made that our MinVar mechanism can successfully defend the federated learning system when there exist multiple attackers. It can also be seen that our MinVar method is capable of defending against more attackers than the Bulyan defense method, which is proposed to only defend up to $m \leq \frac{1}{4}(n-3)$ attackers, while MinVar can defend up to $m \leq \lceil n/2 \rceil - 1$ attackers. With such requirement, besides the difficulties of the task and dimensionality of the model as previously discussed, Bulyan's performance is also largely dependent on the ratio of attackers to total workers (ratio of m:n). When such ratio is small, Bulyan does show the some defending functionality by saving the FL system from divergence to, say about 58% on MNIST as shown in Fig. 6(c) and (e), but with a more difficult task, its improvement could be disregarded as shown in Fig. 7.

Another interesting found is that when the scales of the attack are small, i.e., in our case when scale = 1 and 2, our MinVar defense method still works very well w.r.t. the attack and the Bulyan defense method, however, they are not as well as when the scales are large (4–9). This is especially true when the maximum number of attackers is reached (in 20p9 A, and 15p7 A scenarios, where $m = \lceil n/2 \rceil - 1$). Such found may be a bit against intuition, but it does make sense in our scenario. When the attack scale is small, according to the attack model described in Section 3.2, the attack vectors injected won't differ much from the true gradients from the variance/standard deviation point of view, and hence lead to the difficulties of differentiating the attackers from the honest workers, especially when the number of attackers is just 1 less than the number of honest workers. However, it's not a necessary concern, as our defense method is still able to pick out the suspicious workers and protect the overall federated learning system in such cases.

Figs. 8 and 9 show the learned weights ($\mathbf{h}$) distribution among each peers for the multiple attackers scenarios, and similar observations could be made that the pre-assigned attackers are all given near-zero weights, i.e., the white horizontal stripes.

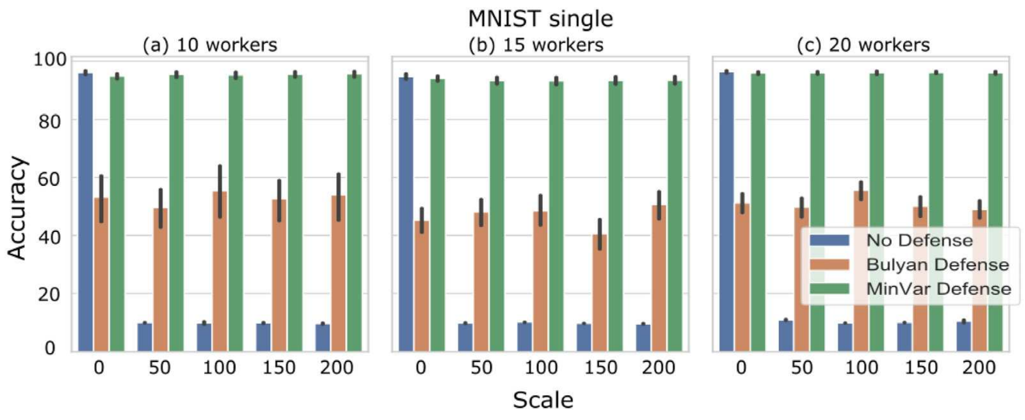

Fig. 2. The performance of MinVar on MNIST when there exists only one attacker in (a) 10 workers, (b) 15 workers, and (c) 20 workers scenarios.

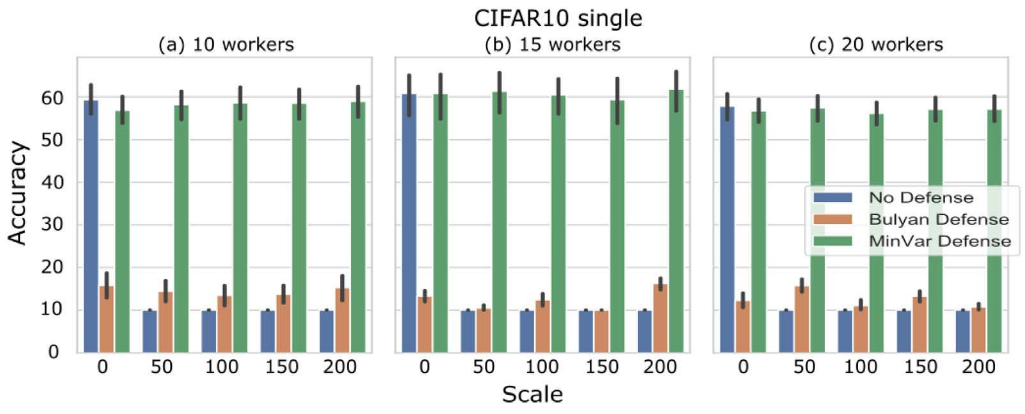

Fig. 3. The performance of MinVar on CIFAR10 when there exists only one attacker in (a) 10 workers, (b) 15 workers, and (c) 20 workers scenarios.

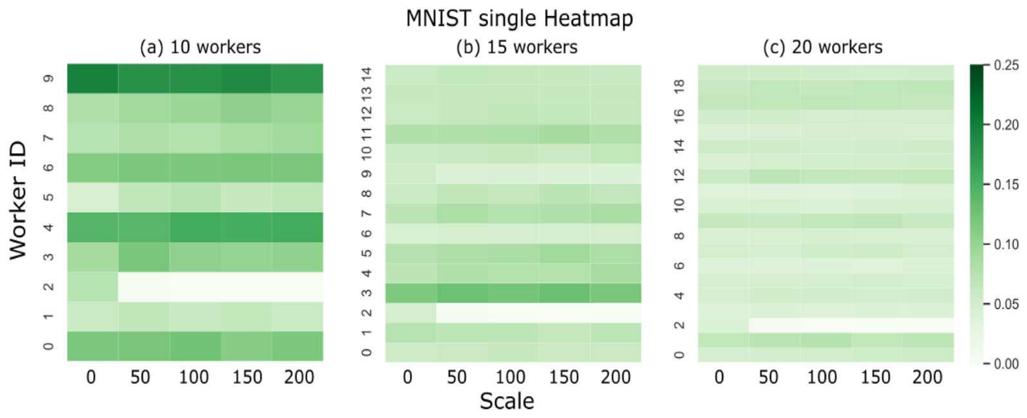

Fig. 4. The heatmap that shows the learned weights(λ) on MNIST when there's only one attacker in the (a) 10 workers, (b) 15 workers, and (c) 20 workers scenarios.

In the previous experiments we compared MinVar with Bulyan, the state-of-the-art defense method for I.I.D. scenarios. Here we compare MinVar with the concatenation of Mini-FL and Bulyan (or Mini-Bulyan) and the concatenation of Mini-FL and Krum (or Mini-Krum). The reason that we choose Mini-Bulyan and Mini-Krum as counterparts comes in two folds: (1) the Mini-FL framework, combined with any existing I.I.D. defense algorithm (e.g. Krum, Median, Bulyan, …etc.), is the newest defense algorithm that consider the non-I.I.D. scenario in FL, and (2) Krum and Bulyan are the state-of-the-art defense algorithms in the I.I.D. scenarios.

Instead of repeating a full-range experiments on the two datasets in both single and multiple attackers scenario, we consider a representative scenario with $n = 20$ and $m = 5$ on the MNIST dataset, as the same setting is also considered in the original paper [30] that proposed the Mini-FL framework. Fig. 10 demonstrates the defense effectiveness of Mini-Bulyan, Mini-Krum and our proposed MinVar algorithms. It could be observed that MinVar outperforms the Mini-Bulyan under all the attack scales, and achieves significantly better defense results than Mini-Krum when the attack scale is large (MinVar and Mini-Krum present comparable defense effectiveness when the attack scale is small). Such results could be caused by a vulnerability of the Mini-FL pre-processing framework. The Mini-FL approach aims to cluster

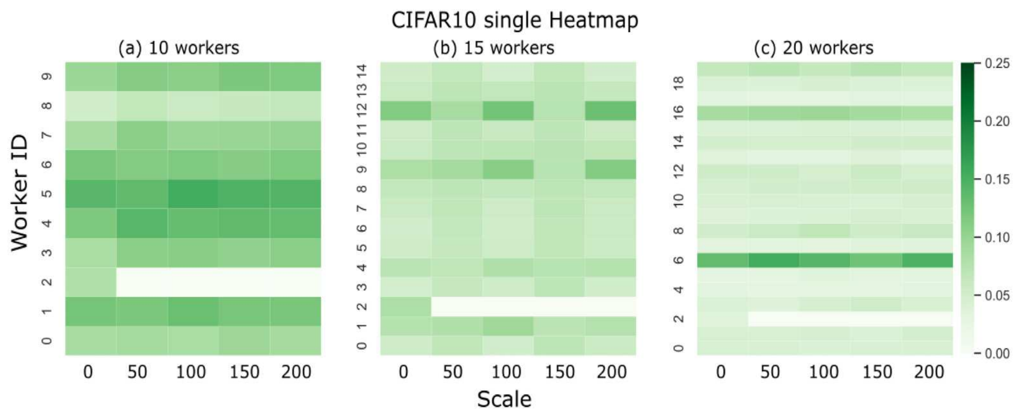

Fig. 5. The heatmap that shows the learned weights(λ) on CIFAR10 when there's only one attacker in the (a) 10 workers, (b) 15 workers, and (c) 20 workers scenarios.

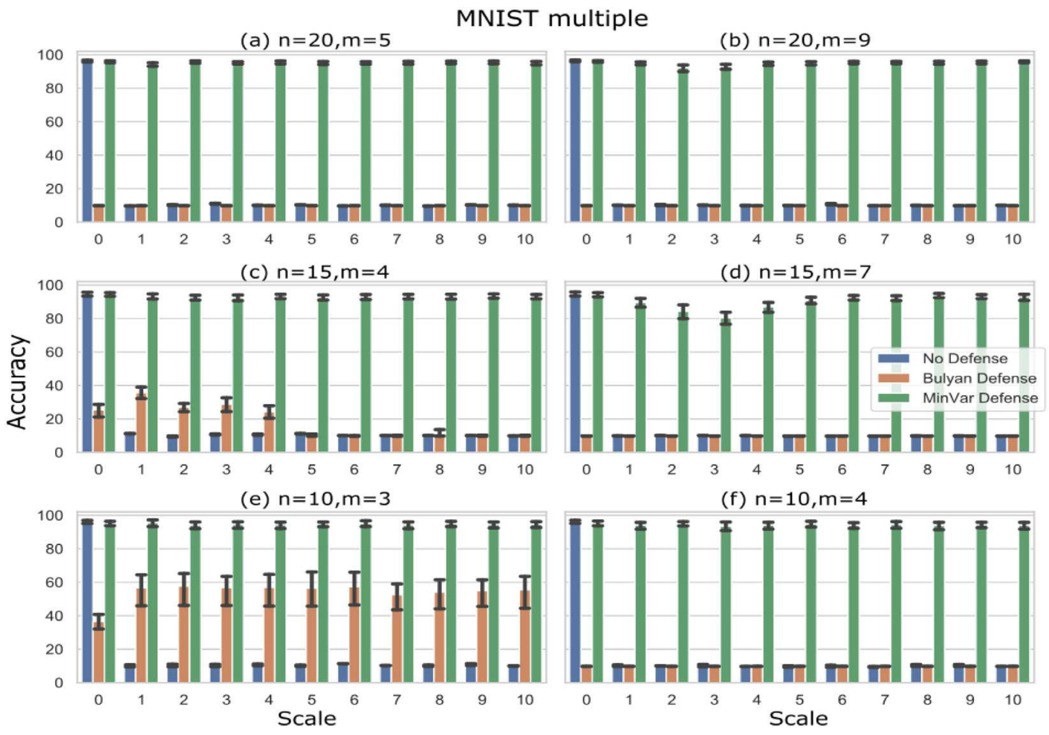

Fig. 6. Defense performance on MNIST when there's are (a) 5 and (b) 9 attackers among 20 workers; (c) 4 and (d) 7 attackers among 15 workers, and (e) 3 and (f) 4 attackers among 10 workers.

"similar" workers' gradients into the same group, and eliminates the outliers in each group by applying the existing I.I.D.-oriented defense methods. During this process, it is highly possible that several similar attackers get clustered into one group. In such case, applying the I.I.D. defense (such as Krum and Bulyan) won't eliminate many of them (if any). When the attack scale is small, eliminating a few attackers may be able to decrease the amount of poisoned gradients injected to the global server in every iteration, leading to the successful defense as in Mini-Krum for scales 0–6. But when the attack scale is large, instead of being suppressed as in the MinVar algorithm, the poisoned gradients will still participate the global model aggregation directly, leading to the failure of FL.

5.4. Validation for the effectiveness of sub-sampling

We take sub-samples of the entire gradients for each local updates and compare the weights (**h**) learned from such sub-samples with the weights learned from the full local updates. 10%, 30%, and 50% of the entire gradients are sampled in the selection of beginning, middle and last training epochs, and the corresponding weights (**h**) learned by our MinVar defense algorithm are shown in Fig. 11. It is suggested that the weights learned by the sub-samples only differ within a very small ranges from the weights trained from the full local updates. And therefore, such small difference of the weights learned from sub-sampling could be disregarded.

We measure the time consumed in each training iteration of the regression model (or the solving iteration of the quadratic problem) in correspondence to the percentage sampled from the full gradients. The result is shown in Fig. 12. Consider the time complexity of the quadratic problem that takes full gradients (100% in Fig. 12) as $\mathcal{O}(k)$, it can be seen that the time consumed by the sub-samples are proportional to the percentage (%) sampled. Such observation confirms our analysis of speed up in Section 4.2.

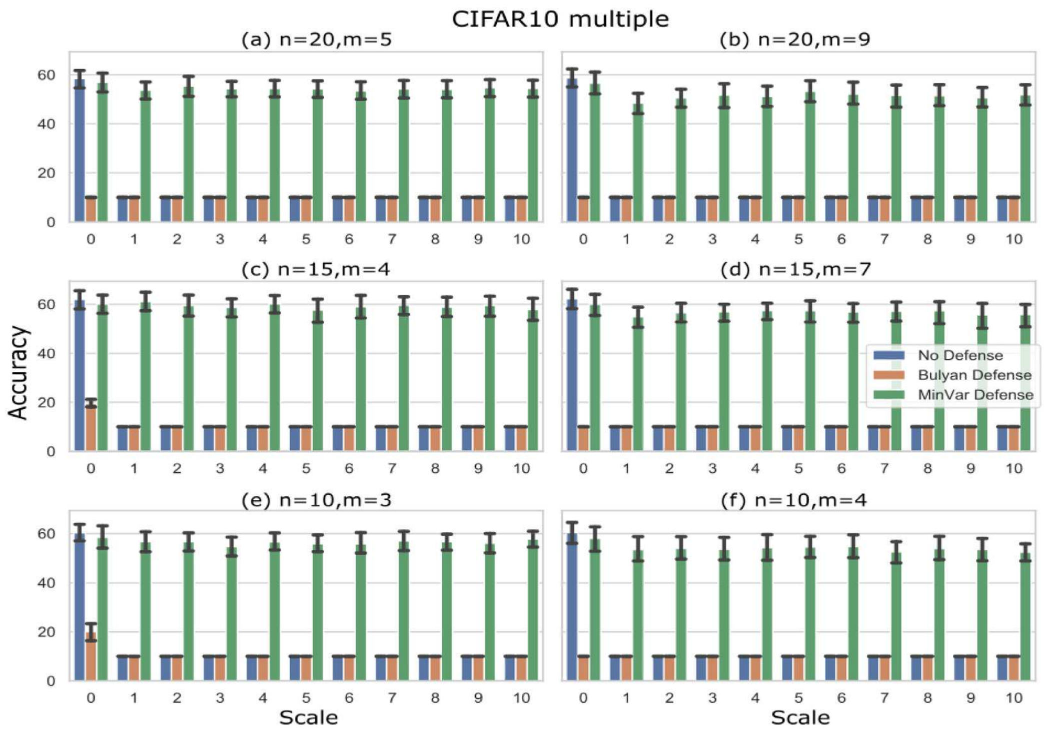

Fig. 7. Defense performance on CIFAR10 when there's are (a) 5 and (b) 9 attackers among 20 workers; (c) 4 and (d) 7 attackers among 15 workers, and (e) 3 and (f) 4 attackers among 10 workers.

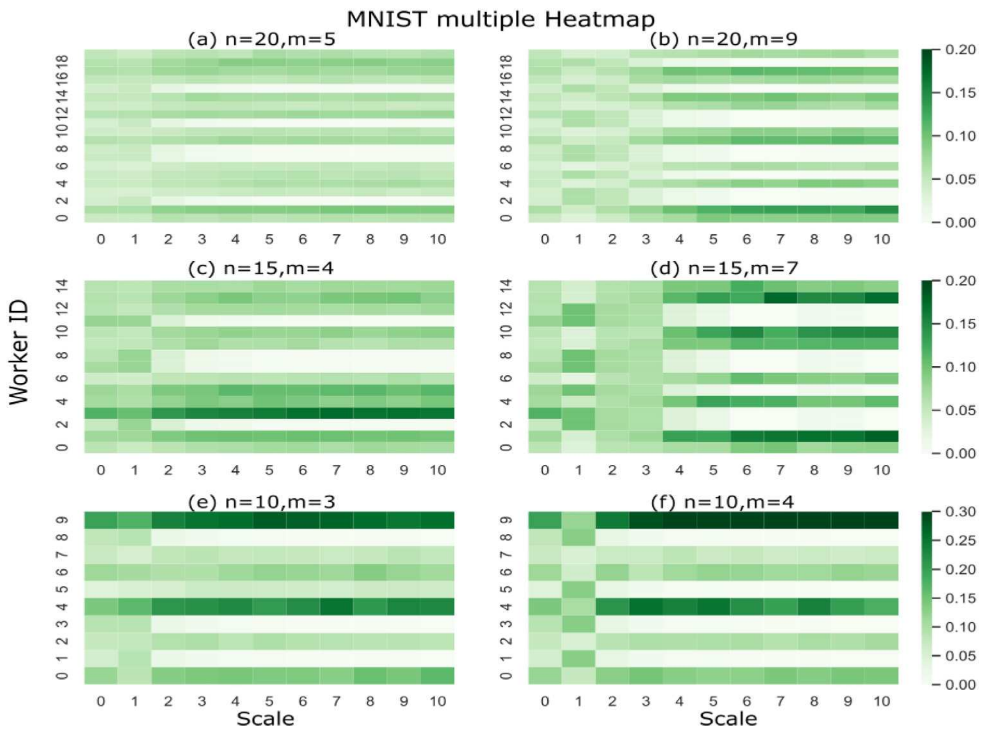

Fig. 8. The heatmap that shows the learned weights (λ) on MNIST when there's are (a) 5 and (b) 9 attackers among the 20 workers; (c) 4 and (d) 7 attackers among the 15 workers, and (e) 3 and (f) 4 attackers among the 10 workers. The number of attackers are picked to disclose the roughly percentage of attackers among the entire worker population.

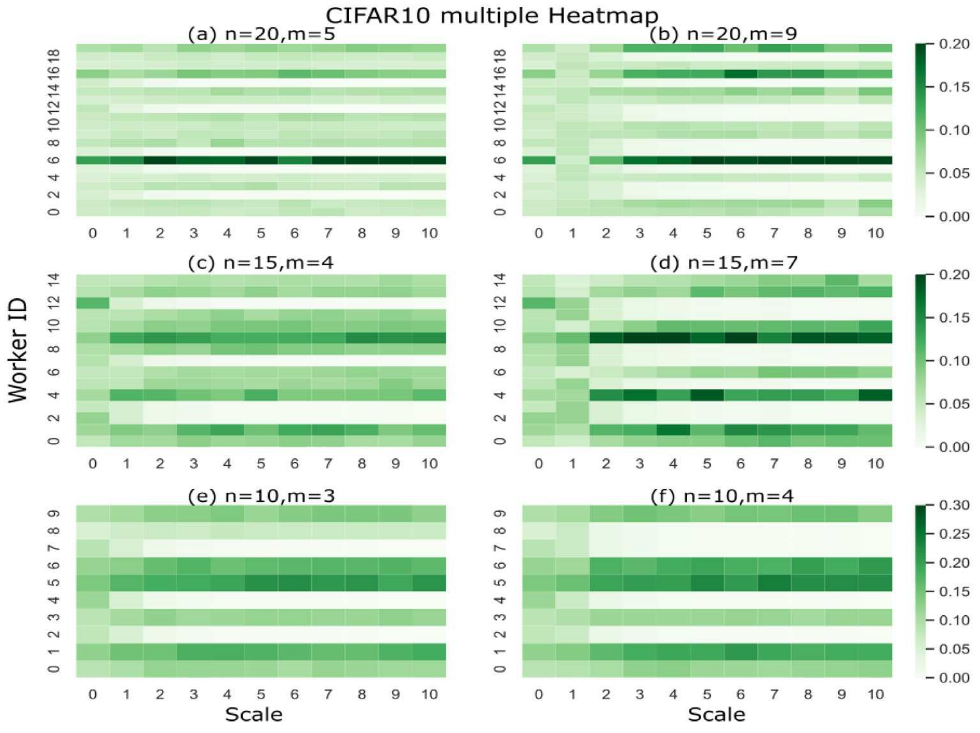

Fig. 9. The heatmap that shows the learned weights (λ) on CIFAR10 when there's are (a) 5 and (b) 9 attackers among the 20 workers; (c) 4 and (d) 7 attackers among the 15 workers, and (e) 3 and (f) 4 attackers among the 10 workers.

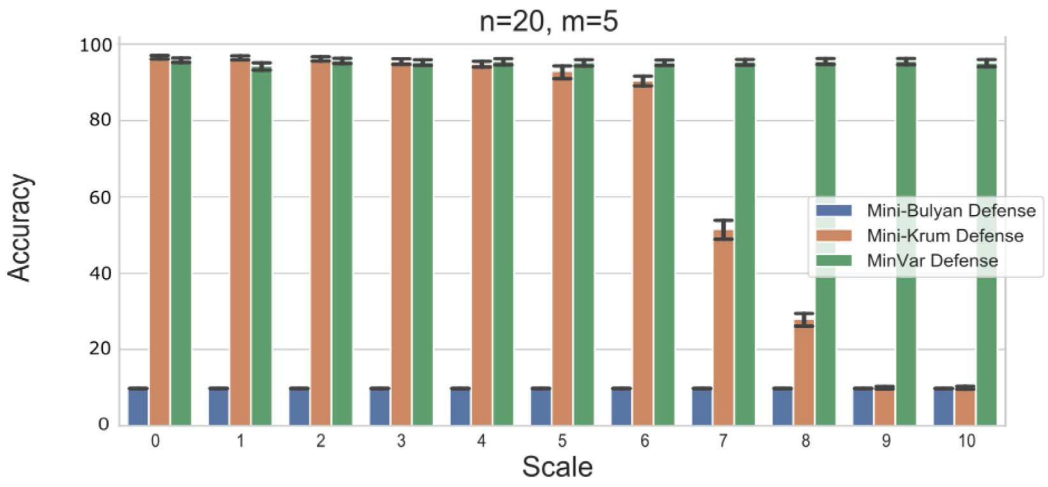

Fig. 10. Defense on MNIST with $n = 20$, $m = 5$.

6. Conclusion

In this paper, we propose a novel defense (robust aggregation) method in federated learning from the variance-minimization approach. The uploaded gradients are utilized as data samples to train a regression model with the objective of minimizing the variance between all local updates, and the regression model outputs the aggregation weights for each local updates. By assigning the learned weights to each worker, our method is capable of ensuring the robust aggregation of non-I.I.D. distributed FL tasks, and meanwhile by assigning close-to-zero weights to suspicious uploaded vectors from potential attackers, our MinVar defense method can minimize the effect of suspicious adversaries with low time consumption.

CRediT authorship contribution statement

Hairuo Xu: Conceptualization, Methodology, Writing – original draft, Formal analysis, Investigation, Software, Validation. **Tao Shu:** Funding acquisition, Supervision, Investigation, Project administration, Resources, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Tao Shu reports financial support was provided by US National Science Foundation. Hairuo Xu reports financial support was provided by US National Science Foundation.

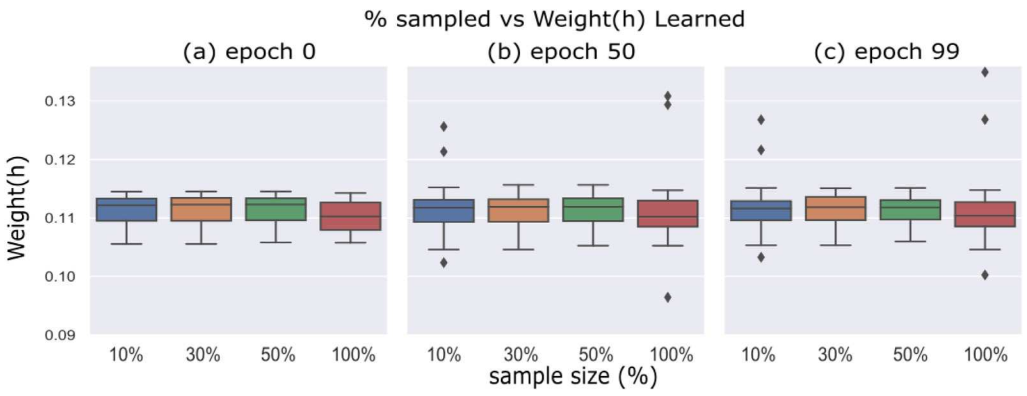

Fig. 11. The comparisons between the weights learned by the sub-samples and learned by full gradients in (a) starting epoch 0, (b) middle epoch 50, and (c) ending epoch 99. The x-axis represent the number of % sampled from the full gradients.

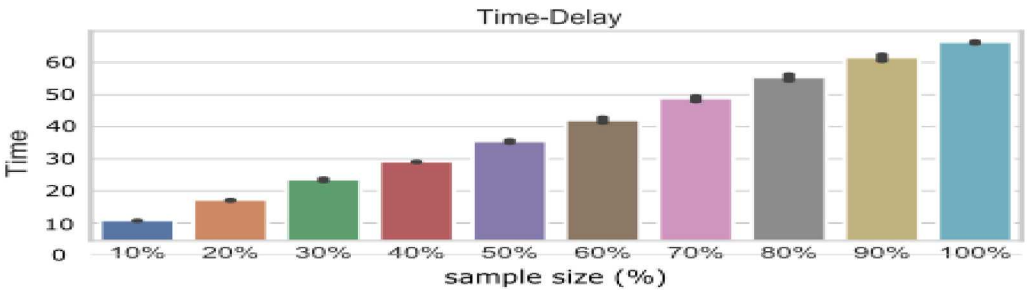

Fig. 12. The time in seconds taken to run the MinVar algorithm on different sub-samples of the full parameters (sub-samples are differentiated by the sampling percentage to the full parameters) in each iteration.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported in part by the United States National Science Foundation (NSF) under grants CNS-2308761 and CNS-2006998. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of NSF.

References

[1] Blanchard P, Mhamdi EME, Guerraoui R, Stainer J. Byzantine-tolerant machine learning. 2017, arXiv preprint arXiv:1703.02757.
[2] Xie C, Koyejo O, Gupta I. Generalized byzantine-tolerant sgd. 2018, arXiv preprint arXiv:1802.10116.
[3] El El Mhamdi M, Guerraoui R, Rouault S. The hidden vulnerability of distributed learning in byzantium. 2018, arXiv e-prints, arXiv–1802.
[4] Yin D, Chen Y, Kannan R, Bartlett P. Byzantine-robust distributed learning: Towards optimal statistical rates. In: International conference on machine learning. PMLR; 2018, p. 5650–9.
[5] Xu H, Shu T. Attack-model-agnostic defense against model poisonings in distributed learning. In: UIC 2022: 19th IEEE international confer- ence on ubiquitous intelligence and computing, haikou, hainan, China, December 15–18. IEEE; 2022.
[6] Fung C, Yoon CJ, Beschastnikh I. Mitigating sybils in federated learning poisoning. 2018, arXiv preprint arXiv:1808.04866.
[7] Tolpegin V, Truex S, Gursoy ME, Liu L. Data poisoning attacks against federated learning systems. In: Computer security–ESORICs 2020: 25th European symposium on research in computer security, ESORICs 2020, guildford, UK, September 14–18, 2020, proceedings, part i 25. Springer; 2020, p. 480–501.
[8] Gu T, Liu K, Dolan-Gavitt B, Garg S. Badnets: Evaluating backdooring attacks on deep neural networks. IEEE Access 2019;7:47230–44.
[9] Chen X, Liu C, Li B, Lu K, Song D. Targeted backdoor attacks on deep learning systems using data poisoning. 2017, arXiv preprint arXiv:1712.05526.
[10] Saha A, Subramanya A, Pirsiavash H. Hidden trigger backdoor attacks. In: Proceedings of the AAAI conference on artificial intelligence, vol. 34, no. 07. 2020, p. 11957–65.
[11] Severi G, Meyer J, Coull S, Oprea A. {Explanation-guided} backdoor poisoning attacks against malware classifiers. In: 30th USeNIX security symposium (USeNIX security 21). 2021, p. 1487–504.
[12] Turner A, Tsipras D, Madry A. Clean-label backdoor attacks. 2018.
[13] Aghakhani H, Meng D, Wang Y-X, Kruegel C, Vigna G. Bullseye polytope: A scalable clean-label poisoning attack with improved transferability. In: 2021 IEEE European symposium on security and privacy (euroS&p). IEEE; 2021, p. 159–78.
[14] Shafahi A, Huang WR, Najibi M, Suciu O, Studer C, Dumitras T, et al. Poison frogs! targeted clean-label poisoning attacks on neural networks. Adv Neural Inf Process Syst 2018;31.
[15] Zhu C, Huang WR, Li H, Taylor G, Studer C, Goldstein T. Transferable clean-label poisoning attacks on deep neural nets. In: International conference on machine learning. PMLR; 2019, p. 7614–23.
[16] Dumford J, Scheirer W. Backdooring convolutional neural networks via targeted weight perturbations. In: 2020 IEEE international joint conference on biometrics. IEEE; 2020, p. 1–9.
[17] Rakin AS, He Z, Fan D. Tbt: Targeted neural network attack with bit trojan. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 13198–207.
[18] Bhagoji AN, Chakraborty S, Mittal P, Calo S. Analyzing federated learning through an adversarial lens. In: International conference on machine learning. PMLR; 2019, p. 634–43.
[19] Fang M, Cao X, Jia J, Gong N. Local model poisoning attacks to {Byzantine-robust} federated learning. In: 29th USeNIX security symposium (USeNIX security 20). 2020, p. 1605–22.
[20] Lyu X, Han Y, Wang W, Liu J, Wang B, Liu J, et al. Poisoning with cerberus: stealthy and colluded backdoor attack against federated learning. In: Thirty-seventh AAAI conference on artificial intelligence. 2023.
[21] Zhou X, Xu M, Wu Y, Zheng N. Deep model poisoning attack on federated learning. Future Internet 2021;13(3):73.
[22] Xie C, Koyejo O, Gupta I. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In: Uncertainty in artificial intelligence. PMLR; 2020, p. 261–70.

[23] Zhang J, Chen J, Wu D, Chen B, Yu S. Poisoning attack in federated learning using generative adversarial nets. In: 2019 18th IEEE international conference on trust, security and privacy in computing and communications/13th IEEE international conference on big data science and engineering (trustCom/bigDataSE). IEEE; 2019, p. 374–80.

[24] Li H, Sun X, Zheng Z. Learning to attack federated learning: A model-based reinforcement learning attack framework. Adv Neural Inf Process Syst 2022;35:35007–20.

[25] Shokri R, Stronati M, Song C, Shmatikov V. Membership inference attacks against machine learning models. In: 2017 IEEE symposium on security and privacy. IEEE; 2017, p. 3–18.

[26] Gu Y, Bai Y, Xu S. CS-MIA: Membership inference attack based on prediction confidence series in federated learning. J Inf Secur Appl 2022;67:103201.

[27] Gu Y, Bai Y. LDIA: Label distribution inference attack against federated learning in edge computing. J Inf Secur Appl 2023;74:103475.

[28] Fu C, Zhang X, Ji S, Chen J, Wu J, Guo S, et al. Label inference attacks against vertical federated learning. In: 31st USeNIX security symposium (USeNIX security 22). 2022, p. 1397–414.

[29] Muñoz-González L, Biggio B, Demontis A, Paudice A, Wongrassamee V, Lupu EC, et al. Towards poisoning of deep learning algorithms with back-gradient optimization. In: Proceedings of the 10th ACM workshop on artificial intelligence and security. 2017, p. 27–38.

[30] Li Y, Yuan D, Sani AS, Bao W. Enhancing federated learning robustness in adversarial environment through clustering non-IID features. Comput Secur 2023;103319.

[31] Park J, Han D-J, Choi M, Moon J. Sageflow: Robust federated learning against both stragglers and adversaries. Adv Neural Inf Process Syst 2021;34:840–51.

[32] Ma Z, Ma J, Miao Y, Li Y, Deng RH. ShieldFL: Mitigating model poisoning attacks in privacy-preserving federated learning. IEEE Trans Inf Forensics Secur 2022;17:1639–54.

[33] Cao X, Fang M, Liu J, Gong NZ. Fltrust: Byzantine-robust federated learning via trust bootstrapping. 2020, arXiv preprint arXiv:2012.13995.

[34] Li M, Andersen DG, Smola AJ, Yu K. Communication efficient distributed machine learning with the parameter server. Adv Neural Inf Process Syst 2014;27.

[35] Baruch G, Baruch M, Goldberg Y. A little is enough: Circumventing defenses for distributed learning. Adv Neural Inf Process Syst 2019;32.

[36] Agrawal A, Verschueren R, Diamond S, Boyd S. A rewriting system for convex optimization problems. J Control Decis 2018;5(1):42–60.

[37] Zinkevich M, Weimer M, Li L, Smola A. Parallelized stochastic gradient descent. Adv Neural Inf Process Syst 2010;23.

[38] Recht B, Re C, Wright S, Niu F. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. Adv Neural Inf Process Syst 2011;24.

[39] Bottou L. Stochastic gradient descent tricks. In: Neural networks: tricks of the trade. Springer; 2012, p. 421–36.

[40] LeCun Y, Cortes C, Burges C. MNIST handwritten digit database. 2010, p. 18, AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist, 2.

[41] Krizhevsky A, Hinton G, et al. Learning multiple layers of features from tiny images. Toronto, ON, Canada; 2009.