

RobustFed: A Truth Inference Approach for Robust Federated Learning

Farnaz Tahmasebian, Jian Lou, and Li Xiong

Emory University, Computer Science
{ftahmas, jian.lou, lxiong}@emory.edu

Abstract. Federated learning is a prominent framework that enables clients (e.g., mobile devices or organizations) to train a collaboratively global model under a central server’s orchestration while keeping local training datasets’ privacy. However, the aggregation step in federated learning is vulnerable to adversarial attacks as the central server cannot manage clients’ behavior. Therefore, the global model’s performance and convergence of the training process will be affected under such attacks. To mitigate this vulnerability issue, we propose a novel robust aggregation algorithm inspired by the truth inference methods in crowdsourcing via incorporating the worker’s reliability into aggregation. We evaluate our solution on three real-world datasets with a variety of machine learning models. Experimental results show that our solution ensures robust federated learning and is resilient to various types of attacks, including noisy data attacks, Byzantine attacks, and label flipping attacks.

Keywords: Federated Learning · Robustness · Adversarial Attack

1 Introduction

Federated learning (FL) has emerged as a promising new collaborative learning framework to build a shared model across multiple clients (e.g., devices or organizations) while keeping the clients’ data private [21,20,1]. The latter is also known as cross-silo FL, which we focus on in this paper. Such a framework is practical and flexible and can be applied in various domains, such as conversational AI and healthcare [21,22,20]. Training a generalizable model for these domains requires a diverse dataset. Accessing and obtaining data from multiple organizations and centralizing them in a third-party service provider can be impractical considering data privacy concerns or regulations. Yet, we still wish to use data across various organizations because a model trained on data from one organization may be subject to bias and poor generalization performance. FL makes it possible to harness the data for joint model training with better generalization performance without the requirement to share raw private local datasets [1].

In a cross-silo FL framework (as shown in Figure 1), there is a semi-honest global coordinating server and several participating clients. The global server controls the learning process and aggregates the model parameters submitted

by clients during multiple communication rounds. The clients train the same model locally using their local datasets. Then, they share their updated local model parameters, not their raw data, with the server, which aggregates all their contributions and broadcasts back the updated global model parameters.

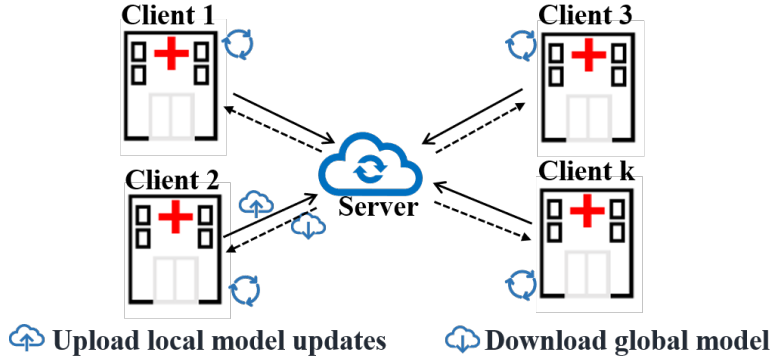


Fig. 1: Overview of Cross-silo Federated Learning (FL) Framework

The most commonly used aggregation algorithm is called Federated Averaging (FedAvg) [21] that takes a weighted average of the local model parameters. This aggregation method is vulnerable to adversarial attacks or unintentional errors in a system. Due to strategic adversarial behavior (e.g., label-flipping and Gaussian noise attacks [14,6,11,4]) or infrastructure failures (e.g., Byzantine faults [17] where client nodes act arbitrarily), the clients can send malicious (manipulated) or arbitrary values to the server. Thus, the global model can be affected severely. Therefore, robust FL against such potential behaviors or failures is essential.

Recently, several methods have been proposed to mitigate attacks in FL or distributed learning [9,5,30,8,4]. The statistical methods such as median or trimmed mean based aggregation (instead of weighted averaging) [30] perform well under Byzantine attack. However, they fail under other types of attacks such as label-flipping and Gaussian noise attacks.

This paper proposes using a truth inference approach for robust aggregation against such attacks in FL. Truth inference is a key component of crowdsourcing that aggregates the answers of the crowd (i.e., workers) to infer the true label of tasks (e.g., traffic incidents, image annotation) [24,15]. We make this connection for the first time that the model parameter aggregation can be formulated as a truth inference problem, i.e., each client is a worker, the local parameters (answers) by the workers need to be aggregated to estimate the global parameter (label). The key idea is to explicitly model the reliability of clients and take them into consideration during aggregation. Such an approach has shown promising results in crowdsourcing compared to simple aggregation approaches such as majority voting (or averaging). However, there are several challenges and

opportunities in applying the truth inference approach for robust FL (compared to crowdsourcing). First, an attacker can manipulate the local training data (e.g., adding noise or flipping the labels) to affect the model parameters (versus directly changing the model parameters). The server only observes the model parameters without access to the data. Hence, a direct application of the truth inference approach on the model parameters cannot detect the malicious clients reliably. Second, FL requires multi-round communication of the local model parameters to the server. This dynamic information creates both challenges and opportunities in detecting unreliable clients. Finally, as in many practical settings, the server does not have access to any golden validation set for validating the local parameter models in order to detect unreliable clients.

To address these challenges, we derive the clients’ reliability score by solving an optimization problem over multiple iterations of FL. We then incorporate the reliability of each client in the aggregation. Our approach is based on two main insights. First, the existing truth inference approaches rely entirely on the derived reliability of the workers for aggregation. In our case, since the model parameters may not accurately reflect the reliability of the workers due to the different kinds of attacks (e.g., label-flipping), we use a pruning algorithm that removes clients with outlier reliability, which mitigates the impact of the malicious clients during aggregation. Second, we exploit the multi-round model parameters submitted by the clients for evaluating the client’s reliability in a more robust way. We briefly summarize our contributions as follows.

- We develop a novel robust aggregation method for FL against potential adversarial attacks and Byzantine failures of clients. The method explicitly models the clients’ reliability based on their submitted local model parameters and incorporates them into aggregation, hence providing a robust estimate of the global model parameters.
- We further enhance the aggregation method by exploiting the multi-round communication of FL and considering the model parameters submitted by the clients both in the previous rounds and the current round for evaluating the client’s reliability.
- We compare our proposed method to several baselines on three image datasets. The results show that our proposed aggregation methods mitigate the impact of attacks and outperform other baselines.

2 Related Works

In this section, we provide a brief review of adversarial attacks on federated learning (FL) along with the existing defense and robustness methods in FL. Subsequently, we briefly review truth inference methods in crowdsourcing.

2.1 Adversarial Attacks on Federated Learning

In federated learning (FL), all the participants agree on a common learning objective and model structure. The attacker aims to compromise the global

model by uploading the malicious data to the global server [21]. The adversary can control the whole local training dataset, local hyper-parameter of a model, and local model parameters in this system.

This paper mainly considers the data poisoning attack scenario, in which malicious clients create poisoned training samples and inject them into their local training dataset [8]. Then, the local model is trained on the dataset contaminated with such poisoned samples. The purpose of this attack is to manipulate the global model to misclassify on test datasets. These attacks can be further divided into two categories: 1) label-flipping attacks [8] and 2) noisy features attack [8]. The label-flipping attack occurs where the labels of training examples of one class are flipped to another class while the data features remain unchanged. For example, an attacker can train a local model with cat images misclassified as a dog and then share the poisoned local model for aggregation. A successful attack forces a model to incorrectly predicts cats to be dogs. In the noisy features attacks, the adversary adds noise to the features while keeping the class label of each data point intact [8]. Noisy data and the backdoor attacks fall in this type of attack [29,28].

FL is vulnerable to poisoning attacks. Studies [8,3] show that just one or two adversarial clients are enough to compromise the performance of the global model. Thus, developing a robust method against these attacks is essential. Fung et al. [8] proposed a defense method, called FoolsGold, against data poisoning attack in FL in a non-IID setting. Their solution differentiates the benign clients from the adversary ones by calculating the similarity of their submitted gradients. Other techniques use the recursive Bayes filtering method [23] to mitigate the data poisoning attack. In some studies [3,25], researchers assume that the global server has access to a golden validation dataset that represents data distribution from clients. The server can detect adversaries by assessing the effectiveness of provided updates on the global model’s performance. If the updates do not improve the global model’s performance, the client is flagged as a potential adversary [3]. However, this method requires the validation dataset which is difficult to achieve in practice.

2.2 Byzantine-Robust Federated Learning

Byzantine clients aim to prevent the global model’s convergence or lead the global model to converge to a poor solution. In some scenarios, the Byzantine clients choose to add Gaussian noise to the gradient estimators, then send these perturbed values to the server. The Byzantine gradients can be hard to distinguish from the benign clients since their variance and magnitude are similar to the benign gradient submissions. Byzantine-Robust methods have been studied in recent years [2,30,23,12,4,18,5]. Most existing methods assume that data is distributed IID among clients and are based on robust statistical aggregation.

A common aggregation method against the Byzantine attack is based on the median of the updates [5]. This method aggregates each model parameter independently. It sorts the local models’ j th parameters and takes the median as the j th parameter for the global model. Trimmed mean [30] is another method that

sorts j th parameters of all local models, then removes the largest and smallest of them, and computes the mean of the remaining parameters as the j th parameter of the global model. Krum [4] selects one of the local models that are similar to other models as the global model. Krum first computes the nearest neighbors to each local model. Then, it calculates the sum of the distance between each client and their closest local models. Finally, select the local model with the smallest sum of distance as the global model. Aggregation methods such as Krum and trimmed mean need to know the upper bound of the number of compromised workers. Other methods extend Krum, such as Multi-Krum [4] and Bulyan [12]. Multi-Krum combines Krum and averaging. Bulyan combines Krum and trimmed mean. It iteratively applies Krum to local models then applies trimmed mean to aggregate the local models.

2.3 Truth Inference Methods

Crowdsourcing aggregates the crowd’s wisdom (i.e., workers) to infer the truth label of tasks in the system, which is called truth inference. Effective truth inference, especially given sparse data, requires assessment of workers’ reliability. There exist various approaches to infer the truth of tasks [13,19,7,27,16,10,32], including direct computing [13], optimization [13,19], probabilistic graphical model (PGM) [7,27,16], and neural network based [31]. The simplest method is majority voting, which works well if all workers provide answers to all of the tasks. However, it fails when data is sparse and workers may be unreliable, as in many practical settings.

Recently, two experimental studies compared state-of-the-art truth inference methods in a “normal” setting and “adversarial” setting [32,26]. The “adversarial” environment is where workers intentionally or strategically manipulate the answers. In the “normal” setting, the study [32] concluded that truth inference methods that utilize a PGM have the best performances in most settings where the type of tasks are binary and single label. The study in the “adversarial” settings [26] focusing on binary tasks showed that neural networks and PGM based methods are generally more robust than other methods for the binary type of tasks. In our FL setting, since we are dealing with model parameters that are numeric and updates that are dense (i.e. a subset of participants submit their model parameters in each round), we use an optimization based truth inference method PM as a baseline method.

3 Preliminaries

3.1 Federated Learning (FL)

The FL framework is important when the participating organizations desire to keep their data private. Instead of sharing data, they share the model parameters to take advantage of a high volume of data with different distributions and improve the model’s generalization. FL consists of K clients and a global server

G . Each client c_i has their own local dataset $\mathbf{D}_i = \{x_1^i, \dots, x_{l_i}^i\}$, where $|D_i| = l_i$. The total number of samples across all the clients is $\sum_{i=1}^K l_i = l$. The goal of FL is to keep the data local and learn a global model with n parameters $w_G \in \mathbb{R}^n$ which minimizes the loss among all samples $D = \bigcup_{i=1}^K D_i$ in the aim that the model generalizes well over the test data \mathbf{D}_{test} .

At each time step t , a random subset from the clients is chosen for synchronous aggregation, i.e. the global server computes the aggregated model, then sends the latest update of the model to all selected clients. Each client $c_i \in K$ uses their local data \mathbf{D}_i to train the model locally and minimize the loss over its own local data. After receiving the latest global model, the clients starts the new round from the global weight vector w_G^t and run model for E epochs with a mini-batch size B . At the end of each round, each client obtains a local weight vector $w_{c_i}^{t+1}$ and computes its local update $\delta_{c_i}^{t+1} = w_{c_i}^{t+1} - w_G^t$, then sends the corresponding local updates to the global server, which updates the model according to a defined aggregation rule. The simplest aggregation rule is a weighted average, i.e., Federated Averaging (FedAvg), and formulated as follow, where $\alpha_i = \frac{l_i}{l}$ and $\sum_{i=1}^K \alpha_i = 1$.

$$w_G^{t+1} = w_G^t + \sum_{i=1}^K \alpha_i \cdot \delta_i^{t+1} \quad (1)$$

3.2 Adversarial Model

We assume any of the clients can be attackers who have full access to the local training data, model structure, learning algorithms, hyperparameters, and model parameters. The adversary’s goal is to ensure the system’s performance degrades or causes the global model to converge to a bad minimum.

In this paper, we mainly consider the data poisoning attack and Byzantine attack. The data poisoning attack is applied in the local training phase and divided into label-flipping and noisy data attacks. In each round, the attacker trains a new local model (based on the global model from the previous round) on the poisoned training data and uploads the new model parameters to the server. Byzantine attack directly changes the model parameters to be uploaded to the server. For the adversarial model, we follow two assumptions: (1) The number of adversaries is less than 50% of whole clients; (2) the data is distributed among the clients in an independent and identically (IID) fashion.

4 Proposed Robust Model Aggregation

We present our proposed robust aggregation method in this section. The key idea is to explicitly model the reliability of clients inspired by truth inference algorithms and take them into consideration during aggregation. We first introduce the truth inference framework and utilize it in FL to estimate the reliability of provided updates by clients in each round. We further improve it by removing the outlier clients before aggregation to address its limitations of correctly

detecting malicious clients in data poisoning attacks. Finally, we incorporate the multi-round historical model parameters submitted by the clients for more robust aggregation. The high-level system model is illustrated in Figure 2. The server comprises two modules: (1) the reliability score calculator; and (2) the aggregator. The server calculates each client’s reliability based on three proposed methods that is improved upon each other.

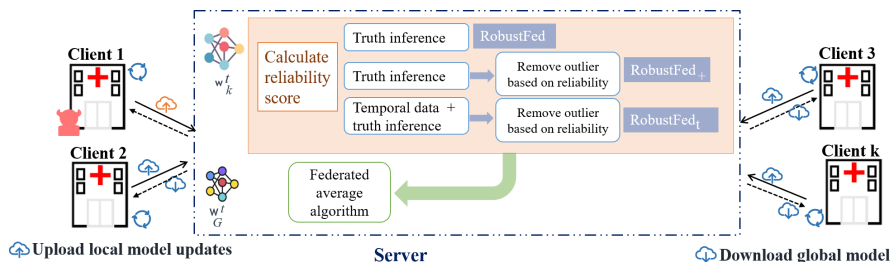


Fig. 2: Overview of Proposed Methods

4.1 Truth Inference Method

Due to the openness of crowdsourcing, the crowd may provide low-quality or even noisy answers. Thus, it is crucial to control crowdsourcing’s quality by assigning each task to multiple workers and aggregating the answers given by different workers to infer each task’s correct response. The goal of truth inference is to determine the true answer based on all the workers’ answers for each task.

Workers Tasks	Observations			Inference			
	w_1	w_2	w_3	Tasks	MV	PM	Ground Truth
t_1	1.72	1.70	1.90	t_1	1.77	1.72	1.72
t_2	1.62	1.61	1.85	t_2	1.69	1.62	1.62
t_3	1.74	1.72	1.65	t_3	1.70	1.74	1.75
t_4	1.72	1.70	1.85	t_4	1.76	1.72	1.71
t_5	1.72	1.71	1.85	t_5	1.76	1.72	1.73

Fig. 3: Example of Crowdsourcing System

Figure 3 shows an example given three workers $\mathbf{W}=\{w_1, w_2, w_3\}$ and five tasks $\mathbf{T}=\{t_1, t_2, \dots, t_5\}$, the goal is to infer the true answer for each tasks. For example, worker w_1 provides 1.72 as an answer to task t_4 . A naive solution to

infer the true answer per task is Majority Voting (MV) or averaging. Based on Figure 3, the truth derived by MV for task t_1 is 1.77, which is inferred incorrectly. A more advanced method such as PM [19] models the reliability of each worker explicitly and resolves conflicts from different sources for each entry. Compared with the ground truth answers, it is clear that worker w_1 and w_2 provide more accurate information (more reliable) while w_3 is not very reliable. By modeling and learning the reliability of workers, PM provides more accurate results compared with averaging.

We can map the model aggregation at the server in FL into the truth inference problem by considering the model’s weight parameters as tasks. In both crowdsourcing and FL, we deal with unlabeled data. In crowdsourcing, the true label of tasks are not available; in FL, the true parameters of the model are unknown (the server does not have access to any validation dataset). The parameter aggregation can be considered as a numeric task (as versus binary task). Algorithm ?? shows the truth inference framework for numeric tasks. The reliability of each worker $i \in [k]$ is denoted as r_{c_i} . It initializes clients’ reliability with the same reliability as $r_{c_i} = 1$. Also, it initializes the estimated truth for each weight parameter as the median of all values provided by the clients. Then it adopts an iterative approach with two steps, 1) inferring the truth and 2) estimating client reliability.

Algorithm 1: Obtain Clients Reliability

Input: Provided parameters by local clients $\delta_k = \bigcup_{i=1}^K \delta_{c_i}, w_G^t$
Output: $\mathbf{R} = \bigcup_{i=1}^K r_{c_i}$

- 1 Initialize clients’ reliability ($r_{c_i} = 1$ for $i \in K$)
- 2 Initialize inferred truth of each update parameter ($\hat{\Delta}_G$) as the median of local updates of δ_k
- 3 **while** *True* **do**
- 4 // Step 1: Inferring the Truth
- 5 **for** each weight parameter $j \in N$ **do**
- 6 | Inferring the $\hat{\Delta}_G$ based on δ_k and \mathbf{R}
- 7 **end**
- 8 // Step 2: Estimating client reliability
- 9 **for** each client **do**
- 10 | estimate \mathbf{R} based on δ_k and $\hat{\Delta}_G$
- 11 **end**
- 12 **if** converge **then**
- 13 | break
- 14 **end**
- 15 **end**

4.2 Robust Aggregation Method: RobustFed

In this section, details of our proposed aggregation method are provided. To begin each round, we compute the reliability level of each client by applying the truth inference method.

Let $\boldsymbol{\delta}_{c_i}^t = \{\boldsymbol{\delta}_{c_i}^t[1], \boldsymbol{\delta}_{c_i}^t[2], \dots, \boldsymbol{\delta}_{c_i}^t[n]\}$ be the local updates that is shared by client c_i at round t . Let $\mathcal{K} = \{c_1, c_2, \dots, c_k\}$ be the set of clients. Hence, at round t , the updated parameters $\boldsymbol{\delta}_k^t$ are collected from K clients. Given the updated parameters $\boldsymbol{\delta}_k^t$ provided by K clients, the goal of utilizing the truth inference is to infer the reliability of each clients $\mathbf{R} = \{r_{c_1}, \dots, r_{c_k}\}$ and incorporate this reliability score into the aggregation method.

The idea is that benign clients provide trustworthy local updates, so the aggregated updates should be close to benign clients' updates. Thus, we should minimize the weighted deviation from the true aggregated parameters where the weight reflects the reliability degree of clients. Based on this principle, we utilize the PM method, which is a truth inference method applicable in numerical tasks [19]. First, by minimizing the objective function, the values for two sets of unknown variables Δ and \mathbf{R} , which correspond to the collection of truths and clients' reliabilities are calculated. The loss function measures the distance between the aggregated parameters (estimated truth) and the parameters provided by client (observation). When the observation deviates from the estimated truth, the loss function return a high value. To constrain the clients' reliabilities into a certain range, the regularization function is defined and it reflects the distributions of clients' reliabilities.

Intuitively, a reliable client is penalized more if their observation is quite different from the estimated truth. In contrast, the observation made by an unreliable client with low reliability is allowed to be further from the truth. To minimize the objective function, the estimated truth relies more on the clients with high reliability. The estimated truth and clients' reliabilities are learned together by optimizing the objective function through a joint procedure. We formulate this problem as an optimization problem as follows:

$$\min_{\mathbf{R}, \hat{\Delta}} \sum_{i=1}^K r_{c_i} \cdot \text{dist}(\hat{\Delta}_G, \boldsymbol{\delta}_{c_i}^t), \quad (2)$$

where r_{c_i} , $\boldsymbol{\delta}_{c_i}^t$ and $\hat{\Delta}_G$ represent client c_i 's reliability, provided update by client c_i at time t , and aggregated updates at time t on the global server, respectively. Also $\text{dist}(\hat{\Delta}_G, \boldsymbol{\delta}_{c_i}^t)$ is a distance function from the aggregated updates of all clients to the clients' provided update. The goal is to minimize the overall weighted distance to the aggregation parameters in the global server in a way that reliable clients have higher weights (importance). In our problem, the type of parameters provided by clients are continuous, therefore Euclidean distance is used as a distance function, $\sqrt{\sum_{j=1}^N (\hat{\Delta}_G^j - \delta_{c_i}^j)^2}$, where N is the number of local parameters and $\delta_{c_i}^j$ indicates the j -th local parameter shared by client c_i . The client c_i 's reliability is modeled using a single value r_{c_i} . Intuitively, workers

with answers deviating from the inferred truth tend to be more malicious. The algorithm iteratively conducts the following two steps, 1) updating the client’s reliability and 2) updating the estimated truth for parameters.

To update the client’s reliability, we fix the values for the truths and compute the clients’ reliability that jointly minimizes the objective function subject to the regularization constraints. Initially, each client is assigned with the same reliability, $\forall_{i \in \mathcal{K}} r_{c_i} = 1$. The reliability score of each client after each iteration is updated as:

$$r_{c_i} = -\log \left(\frac{\sum_{j=1}^N \text{dist}(\hat{\Delta}_G^j, \delta_{c_i}^j)}{\sum_{k'=c_1}^{c_K} \sum_{j=1}^N \text{dist}(\hat{\Delta}_G^j, \delta_{k'}^j)} \right) \quad (3)$$

Equation 3 indicates that a clients reliability is inversely proportional to the difference between its observations and the truths at the log scale.

By fixing the reliability of clients, the truths of parameters are updated in a way that minimizes the difference between the truths and the client’s observations where clients are weighted by their reliabilities and calculated as:

$$\hat{\Delta}_G = \frac{\sum_{i=1}^K r_{c_i} \cdot \delta_{c_i}}{\sum_{i=1}^K r_{c_i}}$$

At the aggregation step, the global server incorporates the provided parameters of each clients based on their reliability. Hence, the global parameters are updated as follows:

$$w_G^{t+1} = w_G^t + \sum_{i \in \mathcal{K}} r_{c_i}^t \cdot \alpha_i \cdot \delta_{c_i}^{t+1} \quad (4)$$

4.3 Reduce Effect of Malicious Clients: RobustFed₊

RobustFed incorporate the reliability of every client in the aggregation but does not include explicit mechanisms to detect and exclude malicious clients. To further reduce the effect of malicious clients, we further propose RobustFed₊ to detect non-reliable clients at each round and discard their participation during the aggregation phase.

Algorithm 2: Robust Aggregation (RobustFed₊)

Input: selected clients K^t , \mathbf{R}^t (reliability of all clients), w_G^t ,
Output: w_G^{t+1}

- 1 **Cand** (set of clients’ candidate) initialized to \emptyset
- 2 $\mathbf{R}^t \leftarrow \text{getClientsReliability}()$
- 3 $\bar{\mu}, \sigma \leftarrow \text{median}(\mathbf{R}^t), \text{std}(\mathbf{R}^t)$
- 4 **for** $i \in K$ **do**
- 5 **if** $\bar{\mu} - \sigma \leq r_{c_i}^t \leq \bar{\mu} + \sigma$ **then**
- 6 Add c_i to **Cand**
- 7 $w_G^{t+1} \leftarrow w_G^t + \sum_{i \in [\mathbf{Cand}]} r_{c_i}^t \cdot \alpha_i \cdot \delta_{c_i}^{t+1}$

Algorithm 2 summarizes RobustFed₊ method. After obtaining the reliability of each clients, the median ($\bar{\mu}$) and standard deviation (σ) of the reliabilities are computed for all the clients participated in the round t . The clients whose reliability fit in the range of $[\bar{\mu} - \sigma, \bar{\mu} + \sigma]$ are selected as a candidate, and the global parameters are updated as follows: $w_G^{t+1} = w_G^t + \sum_{i \in [\mathbf{Cand}]} r_{c_i}^t \cdot \alpha_i \cdot \delta_{c_i}^{t+1}$.

We note that a straightforward method is to remove the clients with lowest reliability scores. Intuitively, we expect the server to assign a higher reliability to honest clients and a lower score to the malicious ones. In our experimental studies, we indeed observe this when no attack happens or under specific types of attacks such as Byzantine or data noise attacks. However, under label-flipping attack, we observe that the RobustFed method assigns higher reliability to the malicious clients. This is because the gradients of the malicious clients can be outliers under such attacks and significantly dominates (biases) the aggregated model parameters, and hence has a high reliability because of its similarity to the aggregated values. Therefore, in our approach, we disregard the clients with reliability deviating significantly from the others.

4.4 Incorporate the Temporal Data to Improve the Defense Capability: RobustFed_t

Given the multi-round communication between the clients and the server in FL, RobustFed and RobustFed₊ only consider one round and ignore the temporal relationship among weight parameters in multiple rounds. Ignoring this temporal relationship might miss important insights of the parameters shared by clients at each rounds. Intuitively, under data poisoning or label flipping attacks, considering the parameters over multiple rounds will more effectively reveal the malicious clients. To take advantage of temporal information, we propose RobustFed_t to incorporate the statistical information of the previous rounds during the reliability estimation. Incorporating the statistical information is dependent on the way the clients are selected in each round:

Static Setting: The server selects the same set of clients at each round to participate in training global model. Therefore, we add the statistics of the model parameters from previous rounds as new tasks in addition to the vector of weights. These statistics are the number of large weights, number of small weights, median of weights and average of weights. The reliability is then evaluated based on all statistics and the parameters submitted in current rounds.

Dynamic Setting: The server dynamically selects a set of clients to join FL and participate in training global model. Since each client may participate with different frequency, we only add median and average of weights from previous round as the weights provided by the new clients.

5 Evaluation

5.1 Experiment Settings

Dataset. We consider the following three public datasets.

- MNIST dataset: This dataset contains 70,000 real-world hand written images with digits from 0 to 9 with 784 features. We split this dataset into a training set and test set with 60,000 and 10,000 samples respectively.
- Fashion-MNIST (fMNIST) dataset: This dataset consists of 28×28 gray scale images of clothing and footwear items with 10 type of classes. The number of features for this dataset is 784. We split this dataset in which training has 60,000 and test data has 10,000 samples.
- CIFAR-10 dataset: This dataset contains 60,000 natural color image of 32×32 pixels in ten object classes with 3,072 features. We split this dataset in which training has 50,000 and test data has 10,000 samples.

For MNIST and fMNIST datasets, we use a 3-layer convolutional neural network with dropout (0.5) as the model architecture. The learning rate and momentum are set as 0.1 and 0.9, respectively. For CIFAR-10, we use VGG-11 as our model. The dropout, learning rate and momentum are set as 0.5, 0.001, 0.9, respectively.

Experiment Setup and Adversarial Attacks. We consider the training data split equally across all clients. For selecting clients to participate in each round, two selection methods are considered, 1) static mode and 2) dynamic mode. In the static mode, the number of clients are set to be 10 and at each iteration, the same set of clients are chosen. In the dynamic mode, the server randomly selects 10 clients from the pool of 100 clients in each round.

We assume that 30% of the clients are adversary. We consider three attack scenarios.

- Label-Flipping Attacks: Adversaries flip the labels of all local training data on one specific class (e.g., class #1) and train their models accordingly.
- Noisy Data: In MNIST and FMNIST, the inputs are normalized to the interval $[0,1]$. In this scenario, for the selected malicious clients, we added uniform noise to all the pixels, so that $x \leftarrow x + U(-1.4,1.4)$. Then we cropped the resulting values back to the interval $[0,1]$.
- Byzantine Attack: Adversary perturb the model updates and send the noisy parameters to the global server. $\delta_i^t \leftarrow \delta_i^t + \epsilon$, where ϵ is a random perturbation drawn from a Gaussian distribution with $\mu = 0$ and $\sigma = 20$.

5.2 Experiment Results

Effect of Attacks on Reliability Score of Clients. Figure 4 shows the reliability range of malicious and benign clients under label-flipping and Byzantine attacks in static mode learned by RobustFed and RobustFed_t, correspondingly. We observe that RobustFed assigns higher reliability to benign workers and vice versa under Byzantine attack and noisy data attack as we expected. However, the opposite behavior is observed under flipping attack. As we discussed, this is likely because the gradients of the malicious clients are outliers under such attacks and significantly dominates (biases) the aggregated model parameters, and hence has high reliability due to the Euclidean distance based evaluation.

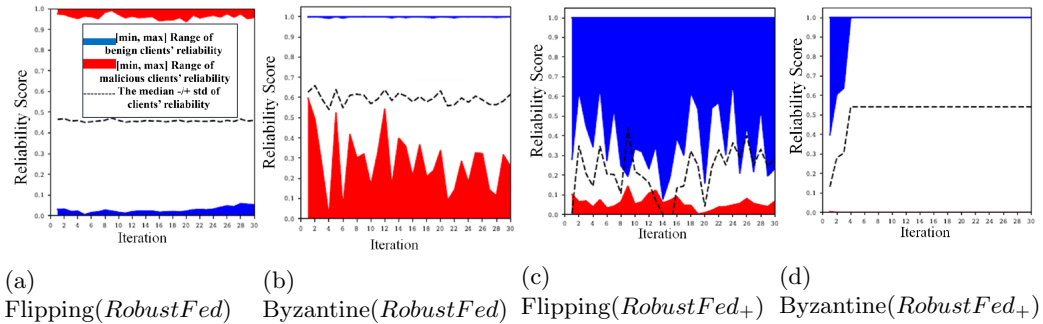


Fig. 4: Range of Clients' Reliability on FMNIST dataset (10 clients, 30% malicious clients)

Therefore, in our Robust₊ approach, we disregard the clients with both high or low reliabilities, which will help mitigate the impact of the malicious clients.

For Robust_t, by incorporating the statistical information of previous rounds, it is able to correctly assign higher reliability to the benign clients (even though with some fluctuations under flipping attacks). It's worth noting that it separates the two types of clients extremely well under Byzantine attack and successfully recognizes malicious clients in all attacks, i.e., assigning close to 0 reliability for them.

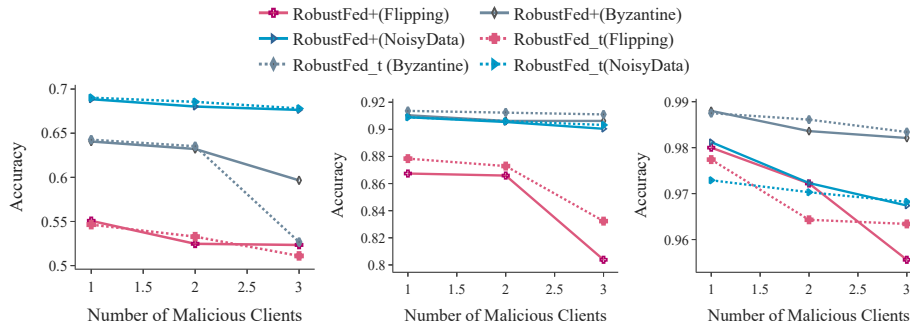


Fig. 5: Effect of number of Malicious Clients

Impact of number of Malicious Clients. We study the impact of the number of malicious clients on the proposed aggregation method. As it is shown in Fig.5, By increasing the number of malicious clients, the performance of the global model slightly drops. It can be observed that RobustFed_t improves upon RobustFed₊ for FMNIST and MNIST datasets that have a higher accuracy on their clean data (i.e., no attack). However, in the CIFAR_10 dataset that has a poor performance on clean data, RobustFed_t could not improve the performance.

Robustness. In this experiment we compare our robust aggregation methods (RobustFed, RobustFed₊, RobustFed_t) with the state-of-the-art baselines. The results of these methods along with average performance are shown in Table 1.

Table 1: Aggregation Method Comparison in Static & Dynamic Mode (30% malicious clients)

Static Mode								
Dataset	Attack	FedAvg	Median	Trim_mean	Krum	RobustFed	RobustFed ₊	RobustFed _t
CIFAR_10	Clean	70.25	70.75	70.78	57.75	68.05	69.74	69.75
	Byzantine	10.0	55.01	10.29	57.24	44.64	59.66	54.67
	Flip Label	51.37	41.34	46.74	10.0	10.0	52.34	51.10
	Noisy	67.51	68.31	68.22	57.67	67.22	67.64	67.80
	Average Performance	42.96	54.88	41.75	41.63	40.62	59.88	58.19
FMNIST	Clean	91.15	90.95	91.05	87.79	91.05	91.05	91.07
	Byzantine	10.0	89.20	10.0	87.66	81.25	90.62	84.59
	Flip Label	79.05	77.58	73.23	10.0	14.55	80.38	83.52
	Noisy	89.25	89.20	89.32	84.78	84.09	87.74	89.0
	Average Performance	59.433	85.32	57.51	60.81	59.96	85.9	85.7
MNIST	Clean	99.29	99.31	99.34	98.51	99.01	99.3	99.32
	Byzantine	11.35	98.18	11.35	97.43	91.35	98.21	98.34
	Flip Label	94.58	97.80	94.47	11.35	11.40	95.56	96.34
	Noisy	92.08	93.01	88.26	83.16	80.04	96.74	96.82
	Average Performance	66	96.33	64.69	63.98	60.93	96.8	97.2
Dynamic Mode								
Dataset	Attack	FedAvg	Median	Trim_mean	Krum	RobustFed	RobustFed ₊	RobustFed _t
CIFAR_10	Clean	69.22	69.58	68.22	56.69	67.87	69.22	67.25
	Byzantine	12.53	44.93	10.00	61.49	55.0	58.78	60.56
	Flip Label	10.0	35.00	10.07	10.32	11.56	57.73	55.53
	Noisy	63.27	63.35	61.18	61.36	61.67	63.43	63.78
	Average Performance	28.6	47.76	27.08	44.39	42.74	59.98	56
FMNIST	Clean	91.68	92.00	88.26	89.79	91.79	91.98	91.87
	Byzantine	10.0	88.90	25.0	90.36	81.35	89.85	83.00
	Flip Label	10.0	68.23	10.25	11.04	11.35	70.93	78.24
	Noisy	89.08	88.12	86.13	81.12	89.24	90.01	90.24
	Average Performance	36.36	81.75	40.46	60.84	60.64	83.49	83.82
MNIST	Clean	99.32	99.35	99.28	99.01	99.32	99.34	99.33
	Byzantine	11.35	97.05	10.01	96.37	96.27	97.07	94.38
	Flip Label	10.28	94.63	10.54	11.35	12.16	94.99	95.23
	Noisy	80.12	96.67	95.34	94.23	87.37	96.10	96.07
	Average Performance	33.91	95.95	38.63	67.31	65.26	96.05	95.22

– **Static Mode.**

In this experiment, clients that participate in each round are fixed. The total number of clients are considered to be 10, in which 30% of them (i.e., 3 clients) are malicious ones. As shown in Table 1, RobustFed₊ and RobustFed_t provide more consistent and better robustness against all three types of attacks while having comparable accuracy on clean data compared with all state-of-the-art methods. As expected, FedAvg’s performance is significantly affected under the presence of malicious clients, especially in Byzantine and flipping attacks. It is also interesting to observe that both Krum and Median are very sensitive to label flipping attacks.

– **Dynamic Mode.** In this experiment, at each round, 10 clients are randomly selected from a pool of 100 clients consists of 30 malicious clients and 70

normal clients. We observe that RobustFed₊ performs stronger robustness by incorporating historical information.

6 Conclusions & Future Works

In this paper, we have studied the vulnerability of the conventional aggregation methods in FL. We proposed a truth inference approach to estimate and incorporate the reliability of each client in the aggregation, which provides a more robust estimate of the global model. In addition, the enhanced approach with historical statistics further improves the robustness. Our experiments on three real-world datasets show that RobustFed₊ and RobustFed_t are robust to malicious clients with label flipping, noisy data, and Byzantine attacks compared to the conventional and state-of-the-art aggregation methods. This study focuses on data with IID distribution among clients; future research could consider non-IID distribution.

References

1. ABADI, M., CHU, A., GOODFELLOW, I., MCMAHAN, H. B., MIRONOV, I., TALWAR, K., AND ZHANG, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016), pp. 308–318.
2. ALISTARH, D., ALLEN-ZHU, Z., AND LI, J. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems* (2018), pp. 4613–4623.
3. BHAGOJI, A. N., CHAKRABORTY, S., MITTAL, P., AND CALO, S. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning* (2019), pp. 634–643.
4. BLANCHARD, P., GUERRAOU, R., STAINER, J., ET AL. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems* (2017), pp. 119–129.
5. CHEN, Y., SU, L., AND XU, J. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 2 (2017), 1–25.
6. DAMASKINOS, G., EL MHAMDI, E. M., GUERRAOU, R., GUIRGUIS, A. H. A., AND ROUAULT, S. L. A. Aggregator: Byzantine machine learning via robust gradient aggregation. In *The Conference on Systems and Machine Learning (SysML), 2019* (2019), no. CONF.
7. DAWID, A. P., AND SKENE, A. M. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics* (1979), 20–28.
8. FUNG, C., YOON, C. J., AND BESCHASTNIKH, I. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866* (2018).
9. FUNG, C., YOON, C. J., AND BESCHASTNIKH, I. The limitations of federated learning in sybil settings. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID) 2020* (2020), pp. 301–316.
10. GAUNT, A., BORSA, D., AND BACHRACH, Y. Training deep neural nets to aggregate crowdsourced responses. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence. AUA Press* (2016), p. 242251.
11. GU, T., LIU, K., DOLAN-GAVITT, B., AND GARG, S. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access* 7 (2019), 47230–47244.
12. GUERRAOU, R., ROUAULT, S., ET AL. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning* (2018), PMLR, pp. 3521–3530.
13. JAGABATHULA, S., SUBRAMANIAN, L., AND VENKATARAMAN, A. Reputation-based worker filtering in crowdsourcing. In *Advances in Neural Information Processing Systems* (2014), pp. 2492–2500.
14. KAIROUZ, P., MCMAHAN, H. B., AVENT, B., BELLET, A., BENNIS, M., BHAGOJI, A. N., BONAWITZ, K., CHARLES, Z., CORMODE, G., CUMMINGS, R., ET AL. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977* (2019).
15. KARGER, D. R., OH, S., AND SHAH, D. Iterative learning for reliable crowdsourcing systems. In *Advances in neural information processing systems* (2011), pp. 1953–1961.
16. KIM, H.-C., AND GHAHRAMANI, Z. Bayesian classifier combination. In *Artificial Intelligence and Statistics* (2012), pp. 619–627.

17. LAMPORT, L., SHOSTAK, R., AND PEASE, M. The byzantine generals problem. In *Concurrency: the Works of Leslie Lamport*. 2019, pp. 203–226.
18. LI, L., XU, W., CHEN, T., GIANNAKIS, G. B., AND LING, Q. Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 1544–1551.
19. LI, Q., LI, Y., GAO, J., ZHAO, B., FAN, W., AND HAN, J. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (2014), ACM, pp. 1187–1198.
20. LIM, W. Y. B., LUONG, N. C., HOANG, D. T., JIAO, Y., LIANG, Y.-C., YANG, Q., NIYATO, D., AND MIAO, C. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials* (2020).
21. MCMAHAN, B., MOORE, E., RAMAGE, D., HAMPSON, S., AND Y ARCAS, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics* (2017), PMLR, pp. 1273–1282.
22. MOTHUKURI, V., PARIZI, R. M., POURIYEH, S., HUANG, Y., DEHGHANTANHA, A., AND SRIVASTAVA, G. A survey on security and privacy of federated learning. *Future Generation Computer Systems* (2020).
23. MUÑOZ-GONZÁLEZ, L., CO, K. T., AND LUPU, E. C. Byzantine-robust federated machine learning through adaptive model averaging. *arXiv preprint arXiv:1909.05125* (2019).
24. RAYKAR, V. C., YU, S., ZHAO, L. H., VALADEZ, G. H., FLORIN, C., BOGONI, L., AND MOY, L. Learning from crowds. *Journal of Machine Learning Research* 11, Apr (2010), 1297–1322.
25. SATTLER, F., WIEDEMANN, S., MÜLLER, K.-R., AND SAMEK, W. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems* 31, 9 (2019), 3400–3413.
26. TAHMASEBIAN, F., XIONG, L., SOTOODEH, M., AND SUNDERAM, V. Crowdsourcing under data poisoning attacks: A comparative study. In *IFIP Annual Conference on Data and Applications Security and Privacy* (2020), Springer, pp. 310–332.
27. VENANZI, M., GUIVER, J., KAZAI, G., KOHLI, P., AND SHOKOUHI, M. Community-based bayesian aggregation models for crowdsourcing. In *Proceedings of the 23rd international conference on World Wide Web* (2014), ACM, pp. 155–164.
28. WANG, B., YAO, Y., SHAN, S., LI, H., VISWANATH, B., ZHENG, H., AND ZHAO, B. Y. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)* (2019), IEEE, pp. 707–723.
29. XIE, C., HUANG, K., CHEN, P.-Y., AND LI, B. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations* (2019).
30. YIN, D., CHEN, Y., RAMCHANDRAN, K., AND BARTLETT, P. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498* (2018).
31. YIN, L., HAN, J., ZHANG, W., AND YU, Y. Aggregating crowd wisdoms with label-aware autoencoders. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (2017), AAAI Press, pp. 1325–1331.
32. ZHENG, Y., LI, G., LI, Y., SHAN, C., AND CHENG, R. Truth inference in crowdsourcing: is the problem solved? *Proceedings of the VLDB Endowment* 10, 5 (2017), 541–552.