RobustFed: A Truth Inference Approach for Robust Federated Learning

Farnaz Tahmasebian
Emory University, Alumni
Atlanta, GA, USA
tahmasebian.farnaz@gmail.com

Jian Lou
Xidian University
Xi'an, China
jlou@xidian.edu.cn

Li Xiong
Emory University
Atlanta, GA, USA
lxiong@emory.edu

ABSTRACT

Federated learning is a prominent framework that enables clients (e.g., mobile devices or organizations) to collaboratively train a global model under a central server's orchestration while keeping local data private. However, the aggregation step in federated learning is vulnerable to adversarial attacks as the central server cannot enforce clients' behavior. As a result, the performance of the global model and convergence of the training process can be affected under such attacks. To mitigate this vulnerability, existing works have proposed robust aggregation methods such as median based aggregation instead of averaging. While they ensure some robustness against Byzantine attacks, they are still vulnerable to label flipping and Gaussian noise attacks. In this paper, we propose a novel robust aggregation algorithm inspired by the truth inference methods in crowdsourcing by incorporating the clients' 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.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**.

KEYWORDS

Federated learning; Robustness; Adversarial attack; Truth discovery

ACM Reference Format:
Farnaz Tahmasebian, Jian Lou, and Li Xiong. 2022. RobustFed: A Truth Inference Approach for Robust Federated Learning. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557439

1 INTRODUCTION

Federated learning (FL) has emerged as a promising collaborative learning framework that builds a shared model across multiple clients (e.g., devices or organizations) while keeping the clients' data private [1, 23, 24]. FL among multiple organizations is also known as cross-silo FL, which we will focus on in this paper. Such a framework can be applied in various domains, such as conversational AI and healthcare [3, 23–25]. 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].

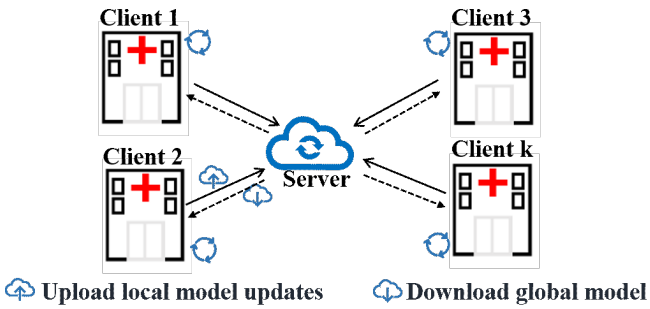


Figure 1: Overview of Cross-silo Federated Learning (FL) Framework

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.

The most commonly used aggregation algorithm is called Federated Averaging (FedAvg) [24] 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 [5, 8, 13, 17, 29]) or infrastructure failures (e.g., Byzantine faults [20] 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. Robust FL against such potential behaviors or failures is essential.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 9 7 8 - 1 - 4 5 0 3 - 9 2 3 6 - 5 / 2 2 / 1 0 …$15.00
https://doi.org/10.1145/3511808.3557439

Recently, several methods have been proposed to mitigate attacks in FL or distributed learning [5, 7, 10, 11, 29, 34]. They are mainly based on statistical robust aggregation such as median or trimmed mean instead of averaging. While they perform well under Byzantine attacks, they fail under other types of attacks such as label-flipping and Gaussian noise attacks.

In this paper, we propose a novel defense method 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) [18, 27]. 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 dataset for validating the local 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 including poisoning attacks and Byzantine attacks on federated learning (FL) along with the existing defense and robustness methods. 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 malicious updates to the global server [24]. The adversary can control the local training dataset, local hyper-parameter of the model, and local model parameters to be uploaded.

In this section, we mainly consider data poisoning attacks, in which malicious clients create poisoned training samples and inject them into their local training dataset [10]. 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 [10] and 2) noisy features attack [10]. 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 labeled as a dog and then share the poisoned local model for aggregation. A successful attack forces a model to incorrectly predict 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 [10]. Noisy data and the backdoor attacks fall in this type of attack [32, 33].

FL is vulnerable to poisoning attacks. Studies [4, 6, 10, 29] 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. [10] 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 [26] to mitigate the data poisoning attack. In some studies [4, 28], 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 improving the global model's performance. If the updates do not improve the global model's performance, the client is flagged as a potential adversary [4]. 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 by uploading modified updates. In some scenarios, the Byzantine clients can 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 by the methods described for data poisoning attacks since their variance and magnitude are similar to the benign gradient submissions. In some other scenarios, the Byzantine clients can also upload random or even adversarially crafted gradient vectors to cause desired attack purposes.

Byzantine-robust methods have been studied in recent years [2, 5–7, 14, 21, 26, 34]. 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 [7]. This method aggregates each model parameter independently. It sorts the local models' jth parameters and takes the median as the jth parameter for the global model. Trimmed mean [34] is another method that sorts jth parameters of all local models, then removes the largest and smallest of them, and computes the mean of the remaining parameters as the jth parameter of the global model. Krum [5] 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, it selects 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 clients. Other methods extend Krum, such as Multi-Krum [5] and Bulyan [14]. 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. [6] leverages the ensemble learning approach to guarantee the security of FL against Byzantine clients, this method significantly increases computational overhead and storage cost.

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. 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. Effective truth inference, especially given sparse data, requires assessment of workers' reliability. There exist various approaches to infer the truth of tasks [9, 12, 16, 19, 22, 31, 36], including direct computing [16], optimization [16, 22], probabilistic graphical model (PGM) [9, 19, 31], and neural network based approaches [35].

Recently, two experimental studies compared state-of-the-art truth inference methods in a "normal" setting and "adversarial" setting [30, 36]. The "adversarial" environment is where workers intentionally or strategically manipulate the answers. In the "normal" setting, the study [36] 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 [30] 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 the underlying method. Based on this experiment study [30], PM method has achieved higher robustness in comparison with the majority voting (MV) in both targeted and untargeted attack scenarios in an "adversarial" environment. We note that our framework is flexible in adopting any truth inference method.

3 PRELIMINARIES

3.1 Federated Learning (FL)

In an FL framework, instead of sharing data, the participating clients share the model parameters to take advantage of the joint data and improve the global model's generalization. FL consists of K clients and a global server G. The same model architecture is shared among the global server and all clients. Each client c_i has their own local dataset $\mathbf{D}_i = \{x_1^i, ...x_{l_i}^i\}$, 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 $\mathbf{D}_{test}$.

In a typical algorithm [24], 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), formulated as follows, 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

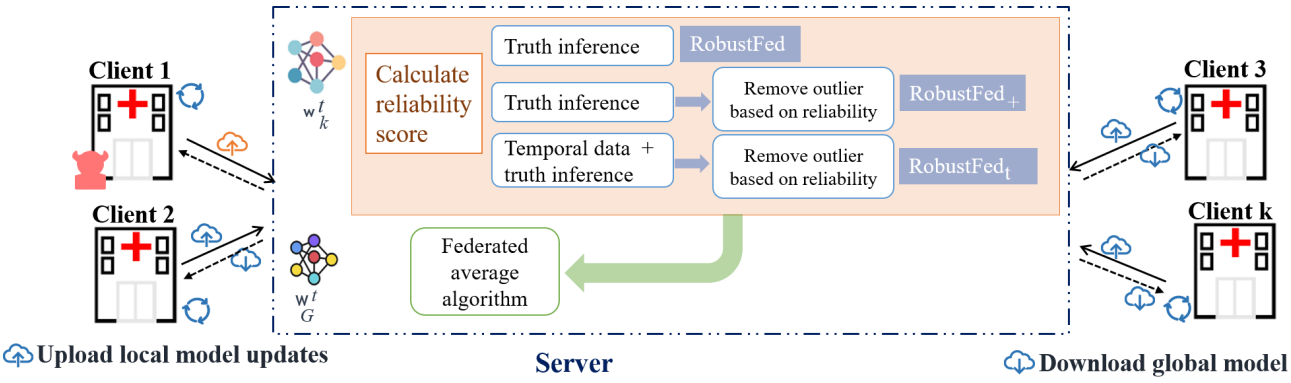


Figure 2: Overview of Proposed Methods

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 entire clients; (2) the data is distributed among the clients in an independent and identically (IID) fashion.

4 ROBUST AGGREGATION BY TRUTH INFERENCE

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 aggregates the client updates based on three proposed methods that are improved upon each other.

4.1 RobustFed Overview

We first explain how we map the model aggregation problem in FL to the truth inference problem in crowdsourcing. Due to the openness of crowdsourcing, the crowd may provide low-quality or even noisy answers. Thus, it is crucial to ensure crowdsourcing's quality by assigning each task to multiple workers and aggregating the answers by different workers to infer each task's correct response. The goal of truth inference is to determine the true answer based on all workers' answers for each task.

Figure 3 shows an example given three workers $\mathbf{W}$=$\{w_1, w_2, w_3\}$ and five tasks $\mathbf{T}$=$\{t_1, t_2, .., t_5\}$, the goal is to infer the true answer for each task. 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 [22] models the reliability of

Workers / Tasks	Observations w_1	w_2	w_3
t_1	**1.72**	**1.70**	**1.90**
t_2	**1.62**	**1.61**	**1.85**
t_3	**1.74**	**1.72**	**1.65**
t_4	**1.72**	**1.70**	**1.85**
t_5	**1.72**	**1.71**	**1.85**

Tasks	Inference MV	PM	Ground Truth
t_1	**1.77**	**1.72**	**1.72**
t_2	**1.69**	**1.62**	**1.62**
t_3	**1.70**	**1.74**	**1.75**
t_4	**1.76**	**1.72**	**1.71**
t_5	**1.76**	**1.72**	**1.73**

Figure 3: Example of Crowdsourcing System

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).

Let $\boldsymbol{\delta}^t_{c_i} = \{\boldsymbol{\delta}^t_{c_i}[1], \boldsymbol{\delta}^t_{c_i}[2], ..., \boldsymbol{\delta}^t_{c_i}[N]\}$ be the local updates shared by client c_i at round t. Let $\mathcal{K} = \{c_1, c_2, ...c_k\}$ be the set of sampled clients at round t. Hence, at round t, the updated parameters $\boldsymbol{\delta}^t_k$ are collected from K clients. Given the updated parameters $\boldsymbol{\delta}^t_k$ provided by K clients, the goal of truth inference is to infer the reliability of each clients $\boldsymbol{R} = \{r_{c_1}, ...r_{c_k}\}$ and incorporate this reliability score into the aggregation method in order to determine the global updates.

Algorithm 1 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: RobustFed

Provided parameters by local clients $\boldsymbol{\delta}_k = \bigcup_{i=1}^{K} \delta_{c_i}, w_G^t$
R=$\bigcup_{i=1}^{K} r_{c_i}$
Initialize clients' reliability ($r_{c_i} = 1 \; for \; i \in K$)
Initialize inferred truth of each update parameter ($\hat{\Delta}_G$) as the median of local updates of $\boldsymbol{\delta}_k$
while *True* **do**
  // Step 1: Inferring the Truth
  for *each weight parameter* $j \in N$ **do**
    Inferring the $\hat{\Delta}_G$ based on $\boldsymbol{\delta}_k$ and **R** using Eq 11
  end
  // Step 2: Estimating client reliability
  for *each client* **do**
    estimate **R** based on $\boldsymbol{\delta}_k$ and $\hat{\Delta}_G$ using Eq 4
  end
  if *converge* **then**
    break
  end
end

4.2 RobustFed Details (Aggregation Method)

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.

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 [22]. First, by minimizing the objective function, the values for two sets of unknown variables Δ and $\boldsymbol{R}$, 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 its 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:

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

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 $dist\ (\hat{\Delta}_G, \boldsymbol{\delta}_{c_i}^t)$ 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, $\sqrt{\sum_{j=1}^{N} \left(\hat{\Delta}_G^j - \boldsymbol{\delta}_{\boldsymbol{c_i}}^j\right)^2}$, 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}.

Each client reliability r_{c_i} is required to be constrained into a certain range, therefore $\zeta(R)$ that reflects the distributions of client's reliability is specified as a regularization function:

$$\zeta(R) = \sum_{i=1}^{K} \exp(-r_{c_i}). \tag{3}$$

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.

Updating Reliability. To update the client's reliability, we fix the values for the truths and compute the clients' reliability that 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:

$$r_{c_i} = -\log\left(\frac{\sum_{j=1}^{N} dist(\hat{\Delta}_G^j \,,\, \delta_{c_i}^j)}{\sum_{k'=c_1}^{c_K} \sum_{j=1}^{N} dist(\hat{\Delta}_G^j \,,\, \delta_{k'}{}^j)}\right). \tag{4}$$

Equation 4 indicates that a client's reliability is inversely proportional to the difference between its observations and the truths at the log scale.

Theorem 4.1. *Suppose that the truths are fixed, the optimization problem Eq 2 with constraint is convex. Furthermore, the global optimal solution is calculated based on Eq 4.*

Proof. Given that the truths are fixed, there is only one set of variable R for the optimization problem Eq 2. Let's define another variable $t_{c_i} = \exp(-r_{c_i})$ and express the optimization problem based on this new variable as follows to prove the convexity of Eq 2:

$$\min_{\boldsymbol{R},\hat{\Delta}} \sum_{i=1}^{K} -\log t_{c_i} \cdot \; dist \; (\hat{\Delta}_G, \boldsymbol{\delta}_{c_i}^t), \quad \text{s.t.} \sum_{i=1}^{K} t_{c_i} = 1. \tag{5}$$

This optimization function is linear combination of negative logarithm functions and thus it is convex. To solve this optimization problem, the method of Lagrange multipliers could be used. The Lagrangian of Eq 5 is given as:

$$\sum_{i=1}^{K} -\log t_{c_i} \cdot \; dist \; (\hat{\Delta}_G, \boldsymbol{\delta}_{c_i}^t) + \lambda(\sum_{i=1}^{K} t_{c_i} - 1), \tag{6}$$

where λ is a Lagrange multiplier. The λ can be derived by setting the partial derivative of Lagrangian with respect to t_{c_i} to be 0,

$$\lambda t_{c_i} = dist(\hat{\Delta}_G\ ,\ \boldsymbol{\delta}^t_{c_i}), \tag{7}$$

and given the constraint that $\sum_{i=1}^{K} t_{c_i} = 1$ we can derive λ as:

$$\lambda = \sum_{k'=c_1}^{c_K} \sum_{j=1}^{N} dist(\hat{\Delta}^j_G\ ,\ {\delta_{k'}}^j). \tag{8}$$

Given that $r_{c_i} = -log(t_{c_i})$, we can derive Eq 4. □

Updating Model Aggregation. 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:

$$\hat{\Delta}_G = \frac{\sum_{i=1}^{K} r_{c_i} \cdot \delta_{c_i}}{\sum_{i=1}^{K} r_{c_i}}. \tag{9}$$

Since the truth (model parameters) are continuous data, the loss function should characterize the distance from the input to the truth with respect to the variance of entries across clients. One common loss function is the normalized squared loss, which is defined as:

$$dist(\hat{\Delta}^j_G\ ,\ {\delta_{k'}}^j) = \frac{(\hat{\Delta}^j_G\ -\ {\delta_{k'}}^j)^2}{std(\delta^j_{c_1}, ..., \delta^j_{c_k})}. \tag{10}$$

Theorem 4.2. *Suppose that the clients' reliabilities are fixed, the optimization problem Eq 2 with Eq 10 is convex. The truth that minimizes the overall weighted distance should be the weighted average of the observations as stated in Eq 9.*

Proof. For proving the convexity, we plug Eq 10 into the objective function Eq 2 and then let the partial derivative with respect to $\hat{\Delta}^j_G$ be 0. Therefore, we can get the optimal truth shown in Eq 9. □

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:

$$w_G^{t+1} = w_G^t + \sum_{i \in K} r^t_{c_i} \cdot \alpha_i \cdot \delta^{t+1}_{c_i}. \tag{11}$$

4.3 Reduce Effect of Malicious Clients: RobustFed$_+$

RobustFed incorporates the reliability of every client in the aggregation but does not include explicit mechanisms to detect and exclude malicious clients. To 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 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 who participated in 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: $w_G^{t+1} = w_G^t + \sum_{i \in [\textbf{Cand}]} r^t_{c_i} \cdot \alpha_i \cdot \delta^{t+1}_{c_i}$.

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.

Algorithm 2: Robust Aggregation (RobustFed$_+$)

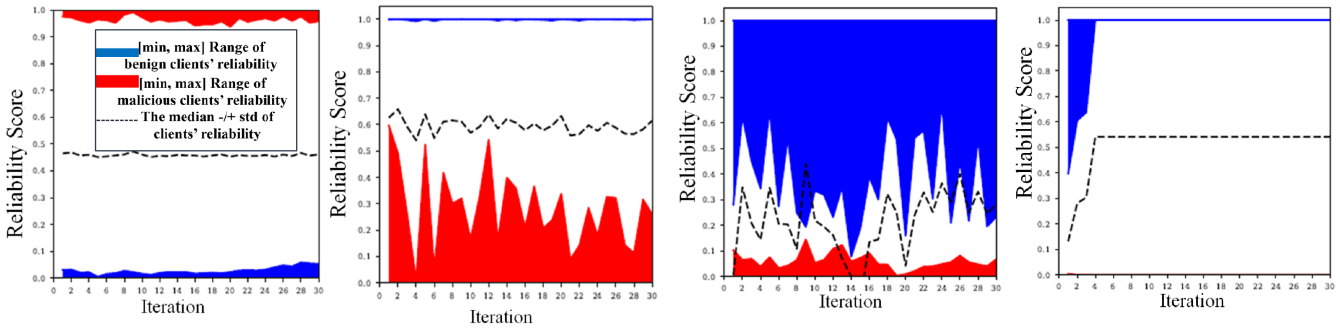
```
selected clients K^t, R^t (reliability of all clients), w_G^t, w_G^{t+1}
Cand (set of clients' candidate) initialized to ∅
R^t ← getClientsReliablity()
μ̄, σ ← median(R^t), std(R^t)
for i ∈ K do
    if μ̄ − σ <= r^t_{c_i} <= μ̄ + σ then
        Add c_i to Cand
    end
end
w_G^{t+1} ← w_G^t + Σ_{i∈[Cand]} r^t_{c_i} · α_i · δ^{t+1}_{c_i}
```

4.4 Incorporate the Multi-round Communication: 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 model parameters in multiple rounds. Ignoring this temporal relationship might miss important information of the parameters shared by clients at each round. Intuitively, under data poisoning or label flipping attacks, considering the parameters over multiple rounds will more effectively reveal 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 described as follows.

Static Setting: The server selects the same set of clients at each round to participate in training the global model. Therefore, we add the statistics of the model parameters (weights) from previous rounds as new tasks in addition to the vector of model parameters of current round. 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 round.

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



(a) **Flipping**($RobustFed$) (b) **Byzantine**($RobustFed$) (c) **Flipping**($RobustFed_+$) (d) **Byzantine**($RobustFed_+$)

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

5 EVALUATION

5.1 Experiment Settings

Datasets. We use the following three public datasets.

- MNIST: 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): this dataset consists of 28×28 gray scale images of clothing and footwear items with 10 types of classes. The number of features for this dataset is 784. We split this dataset into a training set and test set with 60,000 and 10,000 samples respectively.
- CIFAR-10: this dataset contains 60,000 natural color images of 32x32 pixels in ten object classes with 3,072 features. We split this dataset into a training set and a test set with 50,000 and 10,000 samples respectively.

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 [15] as our model. The dropout, learning rate and momentum are set as 0.5, 0.001, 0.9, respectively.

Baseline Aggregation Methods. We consider the following aggregation methods.

- FedAvg [24]: FedAvg computes the average of the clients' local model updates as the global model update, where each client is weighted by its number of training examples.
- Median [34]: Median is a coordinate-wise aggregation rule that considers each model parameter individually. For each model parameter, the server collects its values in all local model updates and sorts them. Median uses the median value of each parameter as the corresponding parameter value in the global model update.
- Trimmed Mean (Trim_mean) [34]: Trimmed mean is another coordinate-wise aggregation rule. The server also sorts the values of each individual parameter in all local model updates. The server removes the largest k and the smallest k values, and then computes the mean of the remaining $n - 2k$ values as the value of the corresponding parameter in the global model update. We set k to be 2.
- Krum [5]: Krum selects one of the n local model updates in each iteration as the global model update based on a smallest Euclidean distance to global model parameters.

Experiment Setup and Adversarial Attacks:. We split the training data 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 by default unless otherwise specified. We consider three attack scenarios.

- Label-Flipping Attacks: Adversaries flip the labels of all local training data of one specific class to another class (e.g., class #1 to #2) and train their models accordingly. All the adversary clients flip labels in the exact same way.
- Noisy Data: In MNIST and FMNIST, the inputs are normalized to the interval [0,1]. 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: Adversaries 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 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

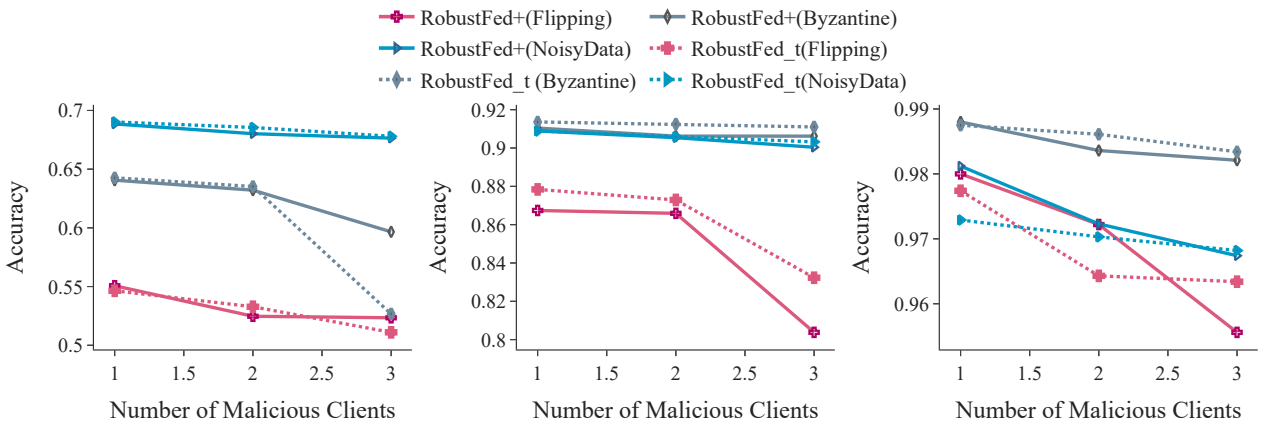


Figure 5: Effect of Number of Malicious Clients on CIFAR_10, FMNIST and MNIST Datasets (Left to Right)

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**
	Robustness	0.14	0.58	0.15	0.17	0.15	**0.75**	0.73
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**
	Robustness	0.11	0.85	0.11	0.11	0.16	0.88	**0.92**
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**
	Robustness	0.11	**0.98**	0.11	0.12	0.12	0.96	**0.97**
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**
	Robustness	0.14	0.5	0.146	0.18	0.17	**0.83**	**0.825**
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**
	Robustness	0.11	0.74	0.12	0.12	0.12	0.77	**0.85**
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
	Robustness	0.1	**0.95**	0.1	0.11	0.12	**0.96**	**0.95**

due to the Euclidean distance based evaluation. Therefore, in our RobustFed$_+$ approach, we disregard the clients with both high and low reliabilities, which will help mitigate the impact of the malicious clients.

For RobustFed$_t$, 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.

Impact of Number of Malicious Clients. We study the impact of the number of malicious clients on the proposed aggregation method. As 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 Comparison. 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. In addition to reporting the accuracy (in percentage) of the different methods on clean data and under different attacks, we also report a robustness metric. The robustness metric is defined as the ratio between the accuracy of the model against the strongest attack (i.e., lowest accuracy) over the accuracy on clean data in the benign setting.

In the **static** mode experiment, clients that participate in each round are fixed. The total number of clients are set to be 10, in which 30% of them (i.e., 3 clients) are malicious. As shown in Table 1, RobustFed$_+$ and RobustFed$_t$ provide more consistent and better robustness against all three types of attacks 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.

Furthermore, the performance of all RobustFed methods maintain accuracy in the benign setting and their performance are comparable to the best accuracy, while other methods like Krum sacrifices the accuracy on clean data.

It can be observed that both RobustFed$_+$ and RobustFed$_t$ methods achieve higher robustness in the CIFAR_10 dataset in comparison with other methods. Given that the MNIST dataset has the highest data quality and CIFAR_10 has the lowest one, the robustness of both RobustFed methods has impacted less in comparison with other methods.

In the **dynamic mode** experiment, at each round, 10 clients are randomly selected from a pool of 100 clients consisting of 30 malicious clients and 70 normal clients. We observe that RobustFed$_+$ has the overall strongest performance by incorporating historical information.

Similar to the static mode, the performance of most of the methods are comparable on clean data except for Krum. We compare the dynamic mode verses static mode, as shown in Table 1, the performances in dynamic mode are slightly lower than static mode since it is more challenging due to the dynamic and sparse participation of the clients.

6 CONCLUSION & FUTURE WORK

In this paper, we 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 more 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.

7 ACKNOWLEDGMENTS

This research has been partially supported by National Science Foundation (NSF) CNS-2124104, CNS-2125530, CNS-1952192, National Institutes of Health (NIH) R01LM013712, Cisco Research #2738379, and Mitsubishi Pharma America #BNI-ALS-005.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. 2018. Byzantine stochastic gradient descent. *Advances in Neural Information Processing Systems* 31 (2018).

[3] Rodolfo Stoffel Antunes, Cristiano André da Costa, Arne Küderle, Imrana Abdullahi Yari, and Björn Eskofier. 2022. Federated Learning for Healthcare: Systematic Review and Architecture Proposal. *ACM Transactions on Intelligent Systems and Technology (TIST)* (2022).

[4] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. 2019. Analyzing Federated Learning through an Adversarial Lens. In *International Conference on Machine Learning*. 634–643.

[5] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems* 30 (2017).

[6] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2021. Provably secure federated learning against malicious clients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 6885–6893.

[7] Yudong Chen, Lili Su, and Jiaming Xu. 2017. 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.

[8] Georgios Damaskinos, El-Mahdi El-Mhamdi, Rachid Guerraoui, Arsany Guirguis, and Sébastien Rouault. 2019. Aggregathor: Byzantine machine learning via robust gradient aggregation. *Proceedings of Machine Learning and Systems* 1 (2019), 81–106.

[9] Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied statistics* (1979), 20–28.

[10] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. 2018. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866* (2018).

[11] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. 2020. The limitations of federated learning in sybil settings. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. 301–316.

[12] Alex Gaunt, Diana Borsa, and Yoram Bachrach. 2016. Training deep neural nets to aggregate crowdsourced responses. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence. AUAI Press*. 242251.

[13] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access* 7 (2019), 47230–47244.

[14] Rachid Guerraoui, Sébastien Rouault, et al. 2018. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*. PMLR, 3521–3530.

[15] Yani Ioannou, Duncan Robertson, Jamie Shotton, Roberto Cipolla, and Antonio Criminisi. [n.d.]. TRAINING CNNS WITH LOW-RANK FILTERS FOR EFFICIENT IMAGE CLASSIFICATION. ([n. d.]).

[16] Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. 2014. Reputation-based worker filtering in crowdsourcing. In *Advances in Neural Information Processing Systems*. 2492–2500.

[17] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.

[18] David R Karger, Sewoong Oh, and Devavrat Shah. 2011. Iterative learning for reliable crowdsourcing systems. In *Advances in neural information processing systems*. 1953–1961.

[19] Hyun-Chul Kim and Zoubin Ghahramani. 2012. Bayesian classifier combination. In *Artificial Intelligence and Statistics*. 619–627.

[20] Leslie Lamport, Robert Shostak, and Marshall Pease. 2019. The Byzantine generals problem. In *Concurrency: the Works of Leslie Lamport*. 203–226.

[21] Liping Li, Wei Xu, Tianyi Chen, Georgios B Giannakis, and Qing Ling. 2019. RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1544–1551.

[22] Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. 2014. 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*. 1187–1198.

[23] Wei Yang Bryan Lim, Sahil Garg, Zehui Xiong, Dusit Niyato, Cyril Leung, Chunyan Miao, and Mohsen Guizani. 2020. Dynamic contract design for federated learning in smart healthcare applications. *IEEE Internet of Things Journal* 8, 23 (2020), 16853–16862.

[24] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[25] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. 2021. A survey on security and privacy of federated learning. *Future Generation Computer Systems* 115 (2021), 619–640.

[26] Luis Muñoz-González, Kenneth T Co, and Emil C Lupu. 2019. Byzantine-Robust Federated Machine Learning through Adaptive Model Averaging. *arXiv preprint arXiv:1909.05125* (2019).

[27] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *Journal of Machine Learning Research* 11, Apr (2010), 1297–1322.

[28] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. 2019. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems* 31, 9 (2019), 3400–3413.

[29] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. 2022. Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning. In *IEEE Symposium on Security and Privacy*.

[30] Farnaz Tahmasebian, Li Xiong, Mani Sotoodeh, and Vaidy Sunderam. 2020. Crowdsourcing under data poisoning attacks: A comparative study. In *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 310–332.

[31] Matteo Venanzi, John Guiver, Gabriella Kazai, Pushmeet Kohli, and Milad Shokouhi. 2014. Community-based bayesian aggregation models for crowdsourcing. In *Proceedings of the 23rd international conference on World Wide Web*. ACM, 155–164.

[32] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[33] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. 2019. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*.

[34] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*. PMLR, 5650–5659.

[35] Li'ang Yin, Jianhua Han, Weinan Zhang, and Yong Yu. 2017. Aggregating crowd wisdoms with label-aware autoencoders. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 1325–1331.

[36] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. 2017. Truth inference in crowdsourcing: is the problem solved? *Proceedings of the VLDB Endowment* 10, 5 (2017), 541–552.