Identifying the Truth of Global Model: A Generic Solution to Defend Against Byzantine and Backdoor Attacks in Federated Learning

Sheldon C. Ebron, Meiying Zhang, and Kan Yang

Dept. of Computer Science, University of Memphis, USA
{sebron, mzhang6, kan.yang}@memphis.edu

Abstract. Federated Learning (FL) enables multiple parties to train machine learning models collaboratively without sharing the raw training data. However, the federated nature of FL enables malicious clients to influence a trained model by injecting error model updates via Byzantine or backdoor attacks. To detect malicious model updates, a typical approach is to measure the distance between each model update and a *ground-truth model update*. To find such *ground-truth model updates*, existing defenses either require a benign root dataset on the server (e.g., FLTrust) or simply use trimmed mean or median as the threshold for clipping (e.g., FLAME). However, such benign root datasets are impractical, and the trimmed mean or median may also eliminate contributions from underrepresented datasets. In this paper, we propose a generic solution, namely FedTruth, to defend against model poisoning attacks in FL, where the *ground-truth model update* (i.e., the global model update) will be estimated among all the model updates with dynamic aggregation weights. Specifically, FedTruth does not have specific assumptions on the benign or malicious data distribution or access to a benign root dataset. Moreover, FedTruth considers the potential contributions from all benign clients. Our empirical results show that FedTruth can reduce the impacts of poisoned model updates against both Byzantine and backdoor attacks, and is also efficient in large-scale FL systems.

Keywords: FedTruth · Byzantine Attack · Backdoor Attack · Robustness · Federated Learning

1 Introduction

In traditional machine learning, training data is usually hosted by a centralized server (cloud server) that runs the learning algorithm or is shared among a set of participating nodes for distributed learning. However, in many applications, data cannot be shared with the cloud or other participating nodes due to privacy or legal restrictions, especially when multiple organizations are involved. Federated Learning (FL) allows multiple parties, such as clients or devices, to collaboratively train machine learning models without sharing raw training data [21]. All selected clients train the global model on their local datasets and send the local model updates to an aggregator. The aggregator then aggregates all the local model updates and sends the new global model to all the clients selected for the next round of training until convergence is reached. The FL framework is suitable for many AI-driven applications where data is sensitive or legally restricted, such as smart healthcare (e.g., cancer prediction [15]), smart transportation (e.g., autonomous driving), smart finance (e.g., fraud detection), and smart life (e.g., surveillance object detection).

However, the federated nature of FL enables malicious clients to influence a trained model by injecting error model updates. For example, adversaries can control a set of clients to launch *Byzantine attacks* [4,6] (i.e., sending arbitrary model updates to make the global model converge to a sub-optimal model), or *backdoor attacks* [1,3,27,29] (i.e., manipulating local model updates to cause the final model to misclassify certain inputs with high confidence).

Towards model poisoning attacks in FL, existing defenses focus on designing robust aggregation rules by:

- *clustering and removing.* This approach identifies malicious model updates by clustering model updates (e.g., Krum [4], AFA [22], FoolsGold [12] and Auror [25]). However, they only work under specific assumptions about the underlying data distribution of malicious clients and benign clients. For example, Krum and Auror assume that the data of benign clients are independent and identically distributed (iid), whereas FoolsGold and AFA assume the benign data are non-iid. Moreover, these defenses cannot detect stealthy attacks (e.g., constraint-and-scale attacks [1]) or adaptive attacks (e.g., Krum attack [9]).
- *clipping and noising.* This approach clips individual weights with a certain threshold and adds random noise to the weights so that the impact of poisoned model updates on the global model can be reduced [1,23]. In [23], the authors propose FLAME, which first applies clustering to filter model updates and then uses clipping and noising with an adaptive clipping threshold and noise level. However, the clipping and noising also eliminate the contributions from benign clients with underrepresented datasets.
- *trimming and averaging.* This approach finds the mean or median of each weight in the remaining model updates after removing some values that are bigger/smaller than some thresholds (trimmed mean or median [31]) or with low frequency (FreqFed [11]). However, the trimmed mean or median can be easily bypassed using adaptive attacks (e.g., Trim attack [9]).
- *adjusting aggregation weights using root data [5].* This approach assigns different weights based on the distance between each model update and the benign model update from the root dataset. However, it requires the aggregator to access the benign root dataset.

Recently, several works [10,13,17] have been proposed to achieve provable Byzantine robustness by integrating variance-reduced algorithms and byzantine-resilient aggregation algorithms. However, they require prior knowledge of the variance of the gradients [13,17] or only focus on existing byzantine-resilient aggregation algorithms.

Motivation: Based on the above-discussed defenses, we have the following observations:

1. Without knowing clients' local datasets or a benign root dataset, it is difficult to determine whether an outlier is a malicious update or a significant contribution from an underrepresented dataset, especially when local datasets are non-iid. It is not a good idea to remove or clip a benign outlier model update with a significant contribution from under-representative data.
2. Only one representative model update is chosen as the global model in many existing Byzantine-resilient aggregation algorithms (e.g., Krum [4], trimmed median [31]), which means the global model is trained with only a single local dataset in each round. In other words, the efforts and contributions of other clients are wasted;
3. Due to various qualities of data and trained local model, it is unfair to treat all the clients equally (e.g., FLAME [23], FreqFed [11]) or evaluate client contributions based on the size of the training dataset (e.g., FedAvg [21]) during the model aggregation.

This paper aims to design a generic solution to defend against model poisoning attacks in FL with the following properties: 1) it does not have specific assumptions on benign or malicious data distribution or accessing to a benign root dataset; 2) it considers potential contributions from all the benign clients (including those with under-representative data); and 3) it reduces the impacts of poisoned model updates from malicious clients. Specifically, we propose a new model aggregation algorithm, namely FedTruth, which enables the aggregator to find the truth among all the received local model updates. The basic idea of FedTruth is inspired by truth discovery mechanisms [19, 20, 24, 32], which are developed to extract the truth among multiple conflicting pieces of data from different sources under the assumption that the source reliability is unknown a priori. *In each round of FedTruth, the global model update (i.e., ground-truth model update) will be computed as a weighted average of all the local model updates with dynamic weights.*

The contributions of this paper are summarized as follows:

– We develop FedTruth, a generic solution to defend against model poisoning attacks in FL. Compared with existing solutions, FedTruth eliminates the assumptions of benign or malicious data distribution and the need to access a benign root dataset.
– We propose a new approach to estimating the *ground-truth model update* among all the model updates with dynamic aggregation weights in each round. Different from the FedAvg [21] (where the aggregation weight is determined by the size of training dataset) or FLAME [23] (where equal aggregation weight is used regardless of the size of training dataset), the aggregation weights in FedTruth are dynamically chosen based on the distances between the estimated truth and local model updates, following the principle that higher weights will be assigned to more reliable clients.

– We extensively evaluate the robustness of our FedTruth against both Byzantine attacks (model-boosting attack, Gaussian noise attack, and local model amplification attack) and backdoor attacks (distributed backdoor attack, edge case attack, projected gradient descent attack) under three attacking strategies (base attack, with model-boosting, and with constrain-and-scaling). The experimental results show that FedTruth can reduce the impacts of poisoned model updates against both Byzantine and backdoor attacks. Moreover, FedTruth works well on both iid and non-iid datasets.
– We further evaluate the efficiency of the FedTruth in terms of the number of iterations to reach FedTruth convergence and the time consumption for two deployments: FedTruth (with entire model updates as inputs) and FedTruth-layer (deploy FedTruth in each layer of the model). The results show that our methods are efficient in large-scale FL systems.

The remainder of this paper is organized as follows: Section 2 presents the problem statement in terms of the system model, threat model, and design goals. Then, we describe the technical overview of our proposed FedTruth, followed by the concrete formulation. Section 4 shows the key experimental results against both Byzantine attacks and Backdoor attacks. Section 5, we describe the related work. Section 6 concludes the paper.

2 Problem Statement

Federated Learning: A general FL system consists of an aggregator and a set of clients S. Let $\mathcal{D}_k$ be the local dataset held by the client k ($k \in S$). The typical FL goal [21] is to learn a model collaboratively without sharing local datasets by solving

$$\min_w F(w) = \sum_{k \in S} a_k \cdot F_k(w),\ \ s.t.\ \sum_{k \in S} a_k = 1\ (a_k \geq 0),$$

where

$$F_k(w) = \frac{1}{n_k} \sum_{j_k=1}^{n_k} f_{j_k}(w; x^{(j_k)}, y^{(j_k)})$$

is the local objective function for a client k with $n_k = |\mathcal{D}_k|$ available samples. a_k is the aggregation weights, which are usually set as $a_k = n_k / \sum_{k \in S} n_k$ (e.g., FedAvg [21]). The FL training process usually contains multiple rounds, and a typical FL round consists of the following steps:

1. *client selection and model update*: a subset of clients S_t is selected, each of which retrieves the current global model w_t from the aggregator.
2. *local training*: each client k trains an updated model $w_t^{(k)}$ with the local dataset $\mathcal{D}_k$ and shares the model update $\Delta_t^{(k)} = w_t - w_t^{(k)}$ to the aggregator.
3. *model aggregation*: the aggregator computes the global model updates as $\Delta_t = \sum_{k \in S_t} a_k \Delta_t^{(k)}$ and update the global model as $w_{t+1} = w_t - \eta \Delta_t$, where η is the server learning rate.

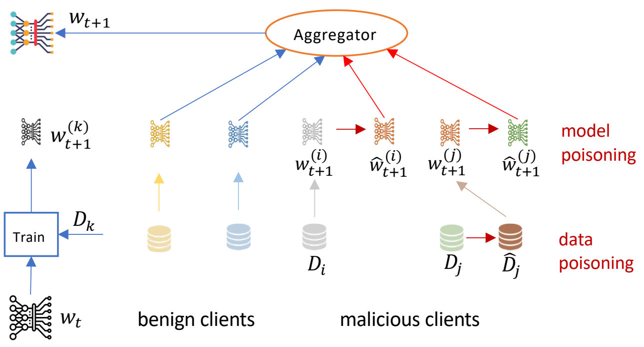

Fig. 1: System Model

FedAvg [21] is the original aggregation rule, which averages all local model weights selected based on the number of samples the participants used. FedAvg has been shown to work well when all the participants are benign clients, but is vulnerable to model poisoning attacks.

System Model: As shown in Figure 1, we consider a typical FL setting, which consists of two entities:

– **Clients**: FL clients are users who participate in the FL process with their end devices, e.g., mobile devices, computers, and vehicles. When selected in an FL round, the clients will train the model based on their local datasets and send local model updates to the aggregator. Due to personal schedules and device status, the group of clients will change dynamically in each FL round. For example, some clients may not be able to send model updates due to low battery or unstable network, and some clients may join the FL task in intermediate FL rounds.
– **Aggregator**: The aggregator is an entity that runs the FL algorithm with the clients, including distributing the initial model to all the selected clients, aggregating local model updates, and sending the global model to the clients selected in a new round.

Threat Model: In this paper, we assume that the aggregator will aggregate all the local model updates honestly in each FL round. However, the clients may be compromised by adversaries and collude to launch Byzantine attacks and backdoor attacks. We assume that the adversaries cannot compromise more than half clients selected in each round. When launching an attack, the adversaries can directly modify their local models (model poisoning attack) and local datasets (data poisoning attack) while having full knowledge about the system (having direct access to any information shared through the system training). However, the adversaries cannot access the benign clients' devices or data. During a Byzantine attack, the adversarial goal is to degrade the global model or prevent it from convergence, while the backdoor attack aims to manipulate the global model by injecting it with a targeted backdoor.

Design Goals: We aim to design a **generic solution** to defend against model poisoning attacks in FL with the following properties: 1) it does not have specific assumptions on benign or malicious data distribution or accessing to a benign root dataset; 2) it considers the potential contributions from all the benign clients (including those with under-representative data); and 3) it reduces the impacts of poisoned model updates from malicious clients.

3 FedTruth: Truth of Global Model

3.1 Technical Overview

In FedAvg [21], the aggregation weight is determined by the size of the training dataset (i.e., $a_k = n_k / \sum_{k=1}^{m} n_k$) where $n_k = |\mathcal{D}_k|$. In some other works, such as FLAME [23], equal aggregation weight is used regardless of the size of the training dataset (i.e., $a_k = 1/m$). However, neither FedAvg nor equal aggregation weights can reflect the performance of a client. In FLTrust [5], the authors proposed using dynamic aggregation weights to calculate the global model. The aggregation weights are estimated based on the trust values, which are calculated based on the similarity between each model update with a *ground-truth model update*. This *ground-truth model update* is trained by the aggregator using a benign root dataset. However, this benign root dataset may not be practical in many applications.

Without a benign root dataset, it is challenging to obtain the *ground-truth model update* among all the local model updates in an FL round. We propose a new model aggregation algorithm, namely FedTruth, which enables the aggregator to uncover the truth among all the received local model updates. The basic idea of FedTruth is inspired by truth discovery mechanisms [19,20,32], which are developed to extract the truth among multiple conflicting pieces of data from different sources under the assumption that the source reliability is unknown a priori. Unlike FLTrust, in FedTruth, we do not obtain the *ground-truth model update* and use it to calculate the aggregated weights. Instead, the *ground-truth model update* is actually the aggregated global model update, which is computed as the weighted average of all the local model updates with dynamic aggregation weights for each round. The aggregation weights in FedTruth are dynamically calculated based on the distances between the estimated truth and local model updates, following the principle that higher weights will be assigned to more reliable clients.

Although the truth discovery approach has been used in RobustFed [26] and *TDFL* [30], they simply apply the *Conflict Resolution on Heterogeneous Data (CRH)* truth discovery algorithm [18] which may still suffer from Byzantine attack or potentially magnifying the local model updates, see the related work for details. Here, we present a generic formulation with a coefficient function and a linear constraint, establishing the necessary conditions for the coefficient function to ensure the convexity and convergence of FedTruth. Furthermore, we demonstrate that our proposed FedTruth is also effective in defending against backdoor attacks, such as the Edge Case [27], DBA [29], and PGD [27] attacks.

3.2 Formulation of FedTruth

Suppose the aggregator receives n_t different model updates $\Delta_t^{(1)}, \cdots, \Delta_t^{(n_t)}$ in FL round t. To find the global update Δ_t^*, we formulate an optimization problem aiming at minimizing the total distance between all the model updates and the estimated global update:

$$\min_{\Delta_t^*, \mathbf{p_t}} D(\Delta_t^*, \mathbf{p_t}) = \sum_{k=1}^{n_t} g(p_t^{(k)}) \cdot d(\Delta_t^*, \Delta_t^{(k)}) \quad s.t. \quad \sum_{k=1}^{n_t} p_t^{(k)} = 1 \tag{1}$$

where $d(\cdot)$ is the distance function and $g(\cdot)$ is a non-negative coefficient function. $p_t^{(k)}$ is the performance of the local model $\Delta_t^{(k)}$ which is calculated based on the distance. Note that, to better understand the performance of each client, our optimization problem is formulated based on the performance values $p_t^{(k)}$ rather than directly on the aggregation weights $a_t^{(k)}$. The aggregation weights can be easily calculated based on the performance value.

There are many different choices of the distance function $d(\cdot)$, e.g., Euclidean distance ($d(\Delta_t^*, \Delta_t^{(k)}) = ||\Delta_t^* - \Delta_t^{(k)}||$) and cosine distance ($d(\Delta_t^*, \Delta_t^{(k)}) = 1 - S_c(\Delta_t^*, \Delta_t^{(k)})$, where S_c is the cosine similarity.

3.3 Solving the optimization problem

We iteratively compute the estimated truth Δ_t^* and the performance values $\mathbf{p^t}$ using coordinate descent approach [2]. Specifically, given an initial global model update Δ_t^* (can be the result of FedAvg or simple average), the algorithm will update the performance values $\mathbf{p^t}$ to minimize the objective distance function. Then, it updates aggregation weight values $a_t^{(k)}$ and uses them to further estimate the new global model update Δ_t^*.

- ***Updating Aggregation Weights:*** Once the truth Δ_t^* is fixed, we first calculate the performance of each model update $\{p_t^{(k)}\}(k = 1, \cdots, n_t)$ as $p_t^{(k)} = d(\Delta_t^*, \Delta_t^{(k)}) / \sum_{k'=1}^{n_t} d(\Delta_t^*, \Delta_t^{(k')})$. Then, the aggregation weights can be updated as
$$a_t^{(k)} = g(p_t^{(k)}) / \sum_{k=1}^{n_t} g(p_t^{(k)}). \tag{2}$$
- ***Updating the Truth:*** Based on the new aggregation weights $\{a_t^{(1)}, \cdots, a_t^{(n_t)}\}$, the truth of global update can be estimated as $\Delta_t^* = \sum_{k=1}^{n_t} a_t^{(k)} \cdot \Delta_t^{(k)}$

The global model update and aggregation weights will be updated iteratively until convergence criteria are met. It is easy to see that the longer the distance between the local model update and the estimated truth, the smaller the aggregation weight will be assigned in calculating the truth. This principle can minimize the impacts of malicious model updates and keep certain contributions from a benign outlier model update.

3.4 Convergence Guarantee of FedTruth

We use the Lagrange multipliers to solve the optimization problem. Under the linear constraint $\sum_{k=1}^{n_t} p_t^{(k)} = 1$, we can define the Lagrangian function of Eq. 1 as

$$L(\{p_t^{(k)}\}_{k=1}^{n_t}, \lambda) = \sum_{k=1}^{n_t} g(p_t^{(k)}) \cdot d(\Delta_t^*, \Delta_t^{(k)}) + \lambda(\sum_{k=1}^{n_t} p_t^{(k)} - 1),$$

where λ is a Lagrange multiplier. To solve the optimization problem, we set the partial derivative with $p_t^{(k)}$ to zero:

$$g'(p_t^{(k)}) \cdot d(\Delta_t^*, \Delta_t^{(k)}) + \lambda = 0 \tag{3}$$

Then, the Eq. 3 can be reformulated as:

$$p_t^{(k)} = g'^{-1}(-\lambda / d(\Delta_t^*, \Delta_t^{(k)})) \tag{4}$$

Since the linear constraint is $\sum_{k=1}^{n_t} p_t^{(k)} = 1$, the λ and $p_t^{(k)}$ can be derived from Eq. 4.

We can see that $g(\cdot)$ must be monotonous and differentiable in the aggregation weight domain in order to guarantee the existence of $g'^{-1}(\cdot)$. Moreover, according to the principle of truth discovery, $g(\cdot)$ should be a decreasing function. Some simple but effective coefficient functions are as follows:

$$g(p_t^{(k)}) = 1/p_t^{(k)} \quad \text{or} \quad g(p_t^{(k)}) = -\log(p_t^{(k)}). \tag{5}$$

Therefore, we say that as long as the coefficient function $g(\cdot)$ is monotonous, decreasing, and differentiable in the aggregation weight domain, the convexity and convergence of FedTruth can be guaranteed. From our experiments, we find that after 5 to 40 iterations of coordinate descent, the estimated truth is close to the converged value.

3.5 Proof of Byzantine-Resilience

In [10], the authors proposed a formal definition of Byzantine-resilience of aggregation algorithm, namely (f, λ)-Resilient Averaging.

Definition 1 ((f, λ)-Resilient Averaging [10]) *For $f < n$ and real value $\lambda > 0$, an aggregation rule F is (f, λ)-Resilient Averaging if for any collection of n vectors $x_1, \cdots, x_n$, and any set $S \subset \{1, \cdots, n\}$ of size $n - f$, the following condition holds*

$$||F(x_1, \cdots, x_n) - \sum_{i \in S} \frac{1}{n-f} x_i|| \leq \lambda \cdot \max_{i,j \in S} ||x_i - x_j||.$$

Under this definition, we show the Byzantine-resilience of FedTruth as in the following theorem:

Theorem 1 *FedTruth is $(f, 1)$-resilient averaging, where $f < n/2$.*

Proof. In FedTruth, the aggregated global model (i,e., the truth) is calculated as the weighted average of all the model updates:

$$FedTruth(x_1, \cdots, x_n) = \sum_{j \in [1,n]} a_j x_j$$

where the aggregation weights a_j are dynamically calculated and $\sum_{j \in [1,n]} a_j = 1$.

For an arbitrary set $S \in \{1, \cdots, n\}$ of size $n - f$, we can rewrite the average of those weights in the set S as

$$\sum_{i \in S} \frac{1}{n-f} x_i = \sum_{i \in S} b_i x_i \quad \text{where} \quad b_i = \frac{1}{n-f} \quad \text{and} \quad \sum_{i \in S} b_i = 1.$$

Then, we can obtain the difference between the truth and the average of set S as

$$\begin{aligned} &||F(x_1, \cdots, x_n) - \sum_{i \in S} \frac{1}{n-f} x_i|| = || \sum_{j \in [1,n]} a_j x_j - \sum_{i \in S} b_i x_i|| \\ &= || \sum_{j \in [1,n]} a_j (x_j - \sum_{i \in S} b_i x_i)|| = || \sum_{j \in [1,n]} a_j \big(\sum_{i \in S} b_i (x_j - x_i) \big)|| \\ &\leq \sum_{j \in [1,n]} a_j \big(\sum_{i \in S} b_i ||x_j - x_i|| \big) \leq \sum_{j \in [1,n]} a_j \big(\sum_{i \in S} b_i \max_{i \in S, j \in [1,n]} ||x_j - x_i|| \big) \end{aligned}$$

Because S is an arbitrary set of size $n - f$, we say that

$$\sum_{i \in S} b_i \max_{i \in S, j \in [1,n]} ||x_j - x_i|| \leq \sum_{i \in S} b_i \max_{i,j \in S} ||x_j - x_i||$$

Then, we have

$$||F(x_1, \cdots, x_n) - \sum_{i \in S} \frac{1}{n-f} x_i|| \leq \sum_{j \in [1,n]} a_j \sum_{i \in S} b_i \max_{i,j \in S} ||x_j - x_i|| = \max_{i,j \in S} ||x_j - x_i||$$

□

3.6 Resisting against Adaptive Attack on FedTruth

In an adaptive attack targeting the Euclidean distance metric, an attacker might be capable of designing an alternative local model, denoted as w^*, such that its Euclidean distance from the baseline model (for instance, the ground truth G) is identical to the Euclidean distance between the actual local model w and the baseline model G. This scenario is feasible if the baseline model remains static and is accessible to the attacker. However, in the FedTruth framework, the baseline model is not a constant; instead, it evolves and is progressively estimated over multiple iterations.

The effectiveness of FedTruth relies on the assumption that the majority of clients are reliable and diverse. If an adversary compromises more than 50% of the clients, they can dominate the results of FedTruth. From our experimental results, we find that when an adversary compromises 40% (4 out of 10) clients in each round, FedTruth can still prevent Byzantine attacks, as seen in Figure 2. However, when the non-iid degree is further increased to 95%, as shown in Figure 5, the accuracy drops and convergence speed becomes slow because some uncompromised clients may perform poorly with highly non-iid training data, leading to a bad estimation of the ground truth by FedTruth. However, our results outperformed all other aggregation algorithms during this experiment, excluding FLTrust.

To counter this, we propose strategies like filtering out inputs from historically unreliable clients, thereby reducing malicious influence. Although FedTruth and FedTruth-layer aim to consider the contributions of all clients, it may be necessary to exclude inputs from clients who have a bad reputation or low reliability in previous FL tasks. To evaluate the reputation or reliability of clients, we need to assess the contributions of each client in an FL task. This motivated us to formulate FedTruth with linear constraints. In practice, we can trim the clients' inputs who have been identified as untrustworthy or unreliable based on their past contributions to FL tasks. By doing so, we can further improve the accuracy and robustness of the global model by preventing the contributions of bad actors from affecting the overall performance. We should also be aware that trimming inputs from clients may have unintended consequences, such as reducing the diversity of the training data and reducing the number of participating clients, potentially leading to overfitting and decreased overall performance. Therefore, we need to carefully evaluate the trade-offs between trimming inputs and maintaining the diversity of the training data. Additionally, leveraging clustering algorithms to categorize model updates before aggregation can help FedTruth remain effective even when faced with a majority of malicious clients.

3.7 Deploying FedTruth in Each Layer

One major challenge in truth discovery is data heterogeneity, which may include non-structured data and missing values. However, this challenge is not applicable to FedTruth because all the local model updates are in the same structure. For example, in deep neural networks, the model updates can be represented as multiple-layer tensors. FedTruth can be run just once by the aggregator to compute the truth of model updates by feeding all the local model updates into the FedTruth algorithm. This deployment treats the local model update as an observed value in the truth discovery algorithms. Also, we can deploy FedTruth on each layer to estimate the truth of that single layer, which means that the weights allocated to all the clients may vary on different layers. We denote this deployment as FedTruth-layer. Such layer-wise deployment seems reasonable, it also brings the computation overhead which is linear to the number of layers. We compare the efficiency between FedTruth and FedTruth-layer in Section 4.6.

4 Experimental Results

We compare the performance of FedTruth with the state-of-the-art aggregation algorithms: FedAvg [21], Krum [4], Trimmed mean [31], Median [31], FLTrust [5] and FLAME [23].

4.1 Experimental Settings

The experimental settings are as follows:

Datasets: We conduct the experiments with MNIST [7], FMNIST [28], and CIFAR-10 [16] datasets. FMNIST and MNIST are comprised of 60,000 black-and-white labeled images of size (28×28). During the *Edge-Case attack* experiment, we used the *Arkiv Digital Sweden* (ARDIS) [17] dataset as the adversarial backdoor images. This dataset is suitable for targeted images when inserting backdoors into MNIST, as ARDIS consists of naturally occurring edge cases.

Clients: When crafting the clients' local datasets, we draw their datapoints randomly from a *non-iid* distribution. We use a *non-iid* distribution because it better represents clients' data in practice than an *iid* distribution. The client's local data is generated a *non-iid* distribution, where the *bias* parameter default is 0.8. In each FL round, we randomly select 10 clients and choose a subset of these selected clients as adversarial clients.

Models: We constructed a Convolutional Neural Network (CNN) classifier for all the experiments considered in this work. It includes an input layer (28x28x1), two convolutional layers with ReLU activation (20x5x1 and 4x4x50), two max pooling layers (2x2), a fully connected layer with ReLU (500 units), and a final fully connected layer with Softmax (10 units). The ResNet-18 model was used when running experiments using the CIFAR-10 dataset. We ran both the MNIST and CIFAR10 experiments on an AWS (g6.xlarge) EC2 instance. When running the FedTruth and FedTruth-layer experiments, we set our convergence threshold to 1e−6.

4.2 Byzantine Attacks

This section presents experimental results for two attacks: the *model-boosting* and *Gaussian noise attacks*, conducted on various FL frameworks. In these attacks, only the model updates (the difference between the newly trained local model and the global model in the previous round) are communicated. For Byzantine experiments performed on the FMNIST and CIFAR-10 datasets, please see our full verion [8].

Model-boosting Attack: The model-boosting attack seeks to degrade the model's performance by boosting the adversary's local updates by a multiple of 10. The subset of compromised clients are randomly selected each round. We conducted experiments with varying percentages of compromised clients in each round to evaluate the robustness of different aggregation algorithms under different attacking scenarios.

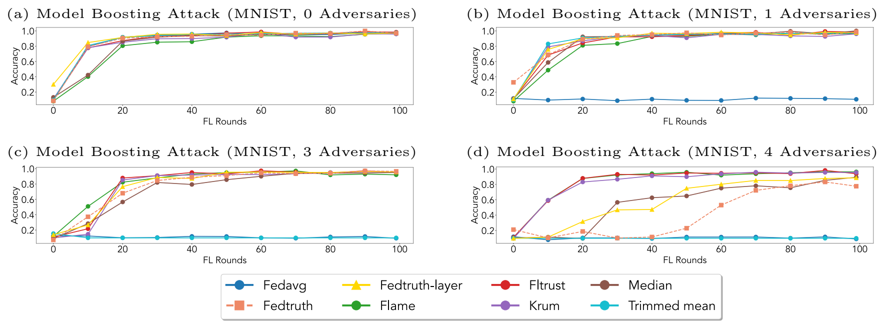

Fig. 2: **Model Boosting Attack** (MNIST, $\times 10$ boosting factor)

Figure 2a shows how all of the aggregation algorithms perform when there are no adversaries present. Figure 2b presents the results when 10% of the clients are compromised in each round. We observe that all of the aggregation algorithms performed well except for FedAvg. However, when the percentage of compromised clients increases to 30%, as shown in Figure 2c, the FedTruth methods remain unaffected by the attack, regardless of the number of adversaries. However, Trimmed-mean, similar to FedAvg, is significantly impacted at this stage and does not reach convergence.

The results in Figure 2d show that increasing the number of adversaries per iteration to 40% slows the convergence rate for the FedTruth, FedTruth-layer, and Median aggregation algorithms. However, they are still able to reach an accuracy of 80% after the 100-th iteration. The FedAvg and Trimmed-mean algorithms were compromised during this version of the experiment as well, preventing them both from reaching any convergence when at least 20% of the clients are adversarial. In contrast, the remaining algorithms (FLTrust, Krum, FLAME) were not affected during this attack regardless of the number of adversaries we selected.

Gaussian Noise Attack: The Gaussian noise attack aims to degrade the performance of the global model by adding Gaussian noise to the model. The noise is drawn from a multivariate Gaussian distribution $N(0, \sigma^2 I)$ [4,5] and is added directly to the adversaries' model before sending it to the aggregator.

In Figure 3a, we show the accuracy and convergence speed of the model for all the aggregation algorithms against the Gaussian noise attack, where three adversaries launch this attack per round. Our findings are as follows: 1) FedTruth can defend against the Gaussian noise attack without significantly slowing down the convergence speed; 2) FedTruth can achieve the same model accuracy as FLTrust, which requires a benign dataset, showing that our FedTruth can actually find the ground truth of the model updates; 3) The convergence speed is impacted in FLAME and Krum, and the model accuracy of FLAME and Krum is not as high as that of FedTruth and FLTrust; and 4) FedAvg cannot converge within 100 rounds under the Gaussian noise attack. In Figure 3b, we combine

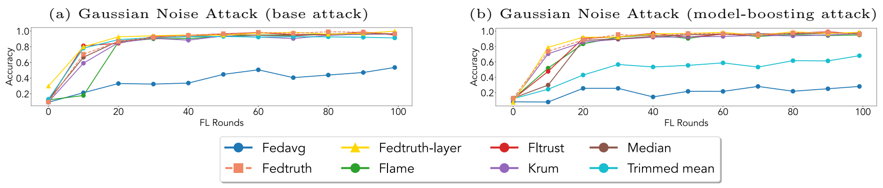

Fig. 3: **Gaussian Noise Attack** (MNIST, 3 adversaries)

the Gaussian noise attack with the model-boosting attack, which also does not degrade the performance of FedTruth or FedTruth-layer. However, FedAvg and Trimmed-mean do not perform well against this attack.

4.3 Backdoor Attacks

This section presents our experimental results for the Distributed Backdoor, Projected Gradient Descent, and Edge Case attacks. During each attack, a trigger was added to all of the adversarial images at the last pixel in the bottom right corner of the image. In our full version [8], we provide results for all attacks combined with the *model-boosting* and *constrain-and-scale* attacks.

Distributed Backdoor Attack (*DBA*): The *DBA* attack distributes a portion of an adversarial background image (*shards*) among all malicious clients who collude with one another each round. The *shards* are then added to the background of the individual adversaries' data.

Figure 4a shows that Krum and FLAME failed to converge on the main task. Therefore, we will disregard their backdoor accuracy, as the models were unable to reach convergence. All of the remaining algorithms were able to reach convergence with a final main task accuracy after 100 rounds at or above 80%. Both of our aggregation algorithms (FedTruth and FedTruth-layer) finished with a final backdoor task accuracy below 10%. FedTruth-Layer does seem to perform slightly better than FedTruth, with the backdoor accuracy never exceeding 20%.

Edge-case Attack: Based on the attacks presented in [27], we implemented a similar attack for the MNIST dataset that used the *Arkiv Digital Sweden (ARDIS) [17]* dataset as the adversarial *edge-cases* that are being injected into the models. During every round of FL, the attackers examined their benign local data points to locate those corresponding to the targeted labels. Using this knowledge, each adversarial client incorporated additional adversarial data points into their training dataset, which represented targeted edge-cases. The number of these added data points was set to be equal to 20% of the matching benign data points.

Figure 4c shows our results for the *main task* accuracy during the *edge-case* attack, and we observed that all algorithms reached convergence with an accuracy that is approximately 90% for all aggregation algorithms after 100 iterations. Figure 4d presents the backdoor accuracy for each of the aggregation

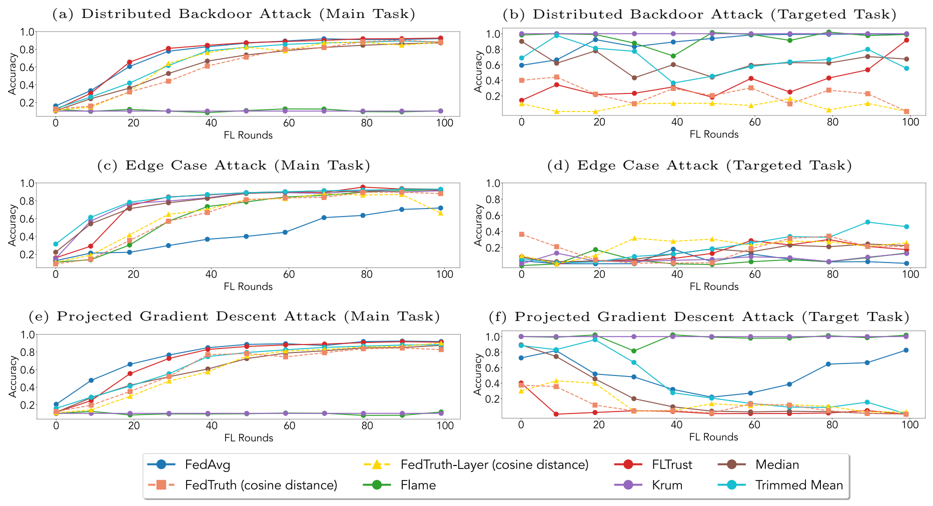

Fig. 4: **Backdoor Attacks** (MNIST, 3 adversaries)

algorithms selected. The final backdoor accuracy for FedTruth and FedTruth-layer are below 10%, similar to Krum and FLAME. Interestingly, FLTrust had a higher backdoor accuracy after 100 iterations, finishing with a backdoor accuracy of 20% close to the final accuracy of FedAvg, which we assume is due to the non-iid sampling being set to 80%. Finally, we observe that the Trimmed mean and Median finish with a backdoor accuracy above 40%.

Projected Gradient Descent Attack (*PGD*): We implemented the *PGD* attacks with the *Torch Attacks* [14] library, which creates a generative model that will take as input an image and returns a perturbed version of the image. We set the max perturbation ($\epsilon = .3$), which is how the adversaries determine how far an image can be noised while generating the adversarial model. Then, we set the step size ($\alpha = .03$) and the number of steps (10).

Figure 4e presents the *main task accuracy* for the *PGD* attack, where we see a decrease in the main task accuracy when using the Krum and FLAME algorithms. The remaining aggregation algorithms are able to train the main task during this attack.

Figure 4f shows the targeted task accuracy during the *PGD* attack. Not considering the algorithms that could not converge on the main task (*Krum* and *FLAME*), we look at the effectiveness of this attack on the remaining algorithms. Trimmed mean allows for the backdoor to be injected in the 20th iteration; however, it does seem to be able to remove the adversarial artifact after the 100th iteration. FedAvg also seems not to be able to remove the backdoor artifact during this attack, as it finishes with a backdoor accuracy of 80%. FedTruth and FedTruth-Layer are able to remove the adversarial artifact after 30 iterations, with the accuracy never exceeding 20% after that round.

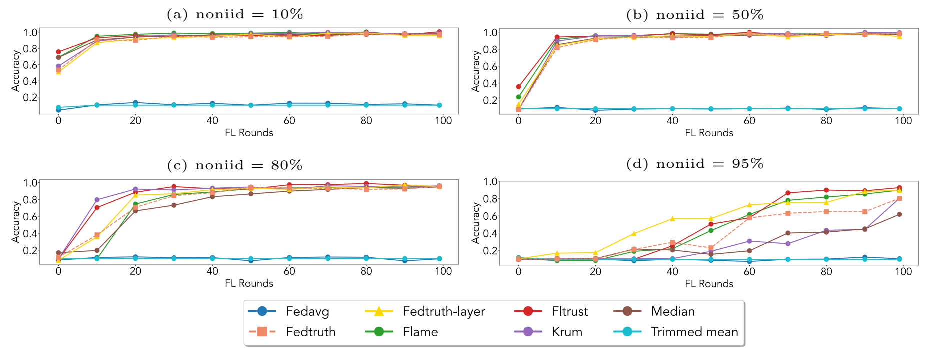

Fig. 5: **Non-iid Impact on Model Boosting Attack** (MNIST, 3 adversaries, ×10 boosting factor)

4.4 The Impact of non-iid on FedTruth

To perform our non-iid experiments, we used label skew, where each of the clients had an equal number of data points. However, each client has a primary label from which the majority of their data points will come. By changing how many data points come from a client's primary, we are able to change the degree to which their data is non-iid.

Figure 5 shows how various non-iid bias parameters affect the experiments when adversaries apply the *model-boosting* attack. During these experiments, three adversaries were selected in each round. Our methodology for sampling non-iid data was specifically engineered to replicate varying degrees of label bias, thus enabling an in-depth analysis of its influence on federated learning model efficacy. We manipulated a bias parameter to adjust the label proportions within each client's local dataset. For instance, setting the bias parameter to 0.9 indicated that 90% of a client's dataset contained instances of their primary label, with the remaining 10% consisting of instances from other labels, allocated based on a Gaussian distribution. For this experiment, we set the bias parameters as 0.1, 0.3, 0.5, 0.8, and 0.95.

The results of the experiments, as seen in Figure 5, suggest that FedTruth can mitigate the impacts of the boosted model regardless of the non-iid degree of the datasets. The FedTruth and FedTruth-layer algorithms do experience some performance degradation as the non-iid degree increases, which is to be expected. However, as seen in Figure 5, after 100 FL iterations, both algorithms reach a top accuracy regardless of the non-iid bias degree.

4.5 Distance Function in FedTruth

From the FedTruth formulation (i.e., Equ. 1), we can see that the distance function plays a significant role in separating benign and malicious model updates. This section discusses how FedTruth performs with different distance functions.

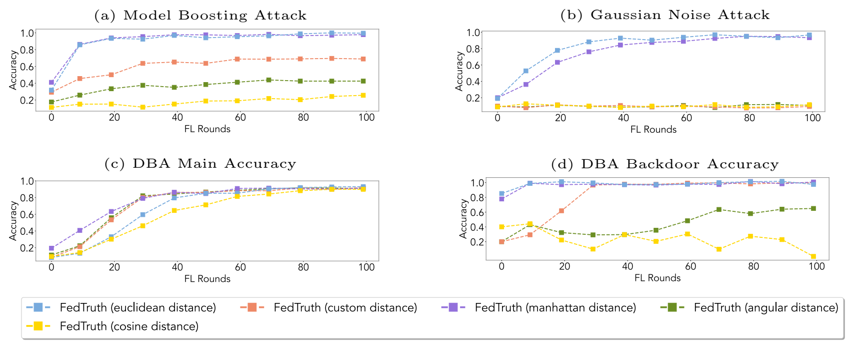

Fig. 6: **Comparison of Distance Functions** (MNIST, 3 adversaries)

Due to page limits, we only present the results of model boosting, Gaussian Noise attack (base attack), and DBA (base attack) here. For more detailed results, please refer to our full version [8].

We evaluate the performance of FedTruth against both Byzantine and backdoor attacks using the following distance functions: 1) two metrics that compute the *difference between the angles* of two vectors (angular distance = arccos(cosine similarity)/π and cosine distance = 1 - cosine similarity); 2) two metrics that determine the *difference between two points* (Euclidean and Manhattan distances); and 3) one custom distance that *combines the angular distance and the Euclidean distance*, which we combine half and half in our results.

Figures 6a and 6b show how the choice of distance measure affects FedTruth and FedTruth-layer during Byzantine attacks. As expected, the two metrics that determine the difference between two points (Euclidean and Manhattan distances) performed the best-reaching convergence in both cases since these approaches can easily determine the adversarial model furthest from the benign ones as they are boosted or have additional random noise inserted into them. However, since the angular distance and cosine distance metrics do not take into consideration the magnitude of a model, it is ineffective during the model boosting attack Figures 6a and since the Gaussian noise attack will only slightly modify the angle, it is not completely effective at removing the adversarial models during this attack.

Figures 6c and 6d present our results for all our distance metrics during a backdoor (DBA) attack. These results indicate that the inverse-cosine similarity or angular distance functions perform the best, as they are able to quickly remove any backdoors injected into the model. However, our main and backdoor accuracy results look good for the models that determine the distance between two points. The custom distance unexpectedly cannot properly train the main task during this experiment.

4.6 Efficiency Evaluation of FedTruth and FedTruth-layer

To compare the computational efficiency, especially in large-scale FL settings, we further evaluate the average aggregation time for 10, 100 and 1000 clients in a single FL round in Table 1, where the aggregation time is calculated as the average of 100 FL rounds. We can see that FedTruth and FedTruth-layer are as efficient as FLTrust (which requires a benign dataset) and much more efficient than FLAME (which also does not require a benign dataset). FLAME becomes very slow when there are 1000 clients in each round. From Table 1, it is easy to see that our FedTruth algorithm can efficiently eliminate the impacts of Byzantine and backdoor attacks, while it does not require a benign root dataset.

Table 1: Comparison of Average Aggregation Time (Model Boosting Attack, 3 Adversaries, MNIST)

Clients per round	**Average Aggregation Time (s)**							
	FedAvg	**FedTruth**	**FedTruth-Layer**	**FLAME**	**FLTrust**	**Krum**	**Median**	**Trimmed Mean**
10	0.006	0.107	0.777	0.194	0.033	4.06	0.041	0.027
100	0.047	0.518	3.498	8.813	0.313	**824.803**	0.173	0.414
1000	1.554	4.903	29.88	**824.549**	3.468	**83172.343**	2.849	6.547

5 Related Work

Defending against model poisoning attacks in federated learning has been an area of active research, with many efforts focusing on designing robust aggregation rules. One approach to identifying and removing malicious model updates involves clustering methods (e.g., Krum [4], AFA [22], FoolsGold [12], and Auror [25]). Although effective, these methods rely on specific assumptions about the underlying data distribution among clients. For example, Krum and Auror assume that benign clients' data are independent and identically distributed (iid), whereas FoolsGold and AFA assume non-iid benign data. Additionally, these defenses may be ineffective against stealthy attacks, such as constraint-and-scale attacks [1], or adaptive attacks, such as the Krum attack [9].

Another approach aims to reduce the impact of poisoned model updates on the global model by clipping individual weights to a certain threshold and adding random noise [1,23]. For instance, FLAME [23] combines clustering with adaptive clipping and noising to mitigate poisoning attacks. However, this technique may unintentionally suppress contributions from benign clients, particularly those with underrepresented datasets. Other methods find the mean or median of model update weights by excluding values based on thresholds (e.g., trimmed mean or median [31]) or frequency of occurrence (FreqFed [11]). Despite their robustness, these approaches are vulnerable to adaptive attacks, such as the Trim attack [9], which exploit these methods' limitations.

Some defenses adjust aggregation weights based on the distance between model updates and a benign root dataset [5]. FLTrust assumes that there is a benign root dataset available to the aggregation server, who will also train and output a server model in each FL round. Upon receiving all the local model updates from clients, the server calculates a *Trust Score* using the ReLU-clipped cosine similarity between each local model update and the server model update. The global model update is computed as the average of the normalized local model updates weighted by the *trust scores*.

In [26], the authors proposed *RobustFed* that applies the truth discovery approach to estimate the reliability of clients in each round. Then, the estimated reliability is used to compute the next round aggregated model. This method suffers from the following two drawbacks. 1) RobustFed applies truth discovery to calculate the reliability $r_{c_i}^t$ of each client c_i in round t, and uses it to aggregate the global model for round $t+1$ (see Eq.11 in RobustFed, $w_G^{t+1} = w_G^t + \sum_{i \in K} r_{c_i}^t \cdot \alpha_i \cdot \delta^{t+1}$). In this case, an attacker can behave honestly to obtain a high reliability score in round t, and launch the Byzantine attack in the next round $t+1$; and 2) Even revising the method to calculate the reliability in the same round, the reliability cannot be directly added to the FedAvg in RobustFed. The reliability is defined by a negative logarithm function of the difference between its local model updates and the truths (ranges between 0 and 1). So, the reliability is a real number ranging between 0 and $+\infty$. The global model aggregation in RobustFed directly adds the reliability on top of the FedAvg, potentially magnifying the local model updates if the reliability is a large number.

TDFL [30] also relies on Truth Discovery to aggregate the global model but, it mainly focuses on applying clustering and clipping filters as shown in FLAME [23] before the truth discovery procedure to defend against Byzantine attacks. However, akin to RobustFed, it simply uses the negative exponential regulation function as detailed in the CRH truth discovery [18]. Recently, several works [10, 13, 17] have been proposed to achieve provable Byzantine robustness by integrating variance-reduced algorithms and byzantine-resilient aggregation algorithms. However, they require prior knowledge of the variance of the gradients [13, 17] or only focus on existing byzantine-resilient aggregation algorithms. In this paper, we propose a generic and robust model aggregation algorithm by computing the aggregation weight dynamically, which is also effective in defending against backdoor attacks, such as DBA [29] and PGD [27].

6 Conclusion

In this paper, we developed FedTruth, a generic solution to defend against model poisoning attacks in FL. Compared with existing solutions, FedTruth eliminates the assumptions of benign or malicious data distribution and the need to access a benign root dataset. Specifically, a new approach was proposed to estimate the *ground-truth model update* (i.e., the global model update) among all the model updates with dynamic aggregation weights in each round, following the principle that higher weights will be assigned to more reliable clients. The experimental results show that FedTruth and FedTruth-layer can efficiently reduce poisoned model updates' impacts against Byzantine and backdoor attacks. Moreover, FedTruth works well on both iid and non-iid datasets.

References

1. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. In: International Conference on Artificial Intelligence and Statistics. pp. 2938–2948. PMLR (2020)
2. Bertsekas, D.P.: Nonlinear programming. Journal of the Operational Research Society **48**(3), 334–334 (1997)
3. Bhagoji, A.N., Chakraborty, S., Mittal, P., Calo, S.: Analyzing federated learning through an adversarial lens. In: International Conference on Machine Learning. pp. 634–643. PMLR (2019)
4. Blanchard, P., El Mhamdi, E.M., Guerraoui, R., Stainer, J.: Machine learning with adversaries: Byzantine tolerant gradient descent. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 118–128 (2017)
5. Cao, X., Fang, M., Liu, J., Gong, N.: Fltrust: Byzantine-robust federated learning via trust bootstrapping. In: Proceedings of NDSS (2021)
6. Chen, Y., Su, L., 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), 1–25 (2017)
7. Deng, L.: The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine **29**(6), 141–142 (2012). `https://doi.org/10.1109/MSP.2012.2211477`
8. Ebron, S.C., Zhang, M., Yang, K.: Identifying the truth of global model: A generic solution to defend against byzantine and backdoor attacks in federated learning (full version) (2025), `https://arxiv.org/abs/2311.10248`
9. Fang, M., Cao, X., Jia, J., Gong, N.: Local model poisoning attacks to byzantine-robust federated learning. In: 29th USENIX Security Symposium. pp. 1605–1622 (2020)
10. Farhadkhani, S., Guerraoui, R., Gupta, N., Pinot, R., Stephan, J.: Byzantine machine learning made easy by resilient averaging of momentums. In: International Conference on Machine Learning. pp. 6246–6283. PMLR (2022)
11. Fereidooni, H., Pegoraro, A., Rieger, P., Dmitrienko, A., Sadeghi, A.R.: Freqfed: A frequency analysis-based approach for mitigating poisoning attacks in federated learning. NDSS (2024)
12. Fung, C., Yoon, C.J., Beschastnikh, I.: The limitations of federated learning in sybil settings. In: 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020). pp. 301–316 (2020)
13. Gorbunov, E., Horváth, S., Richtárik, P., Gidel, G.: Variance reduction is an antidote to byzantine workers: Better rates, weaker assumptions and communication compression as a cherry on the top. In: International Conference on Learning Representations, ICLR (2023)
14. Kim, H.: Torchattacks: A pytorch repository for adversarial attacks. arXiv preprint arXiv:2010.01950 (2020)
15. Kourou, K., Exarchos, T.P., Exarchos, K.P., Karamouzis, M.V., Fotiadis, D.I.: Machine learning applications in cancer prognosis and prediction. Computational and structural biotechnology journal **13**, 8–17 (2015)
16. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
17. Kusetogullari, H., Yavariabdi, A., Cheddad, A., Grahn, H., Hall, J.: Ardis: a swedish historical handwritten digit dataset. Neural Computing and Applications **32**(21), 16505–16518 (2020)

18. Li, Q., Li, Y., Gao, J., Zhao, B., Fan, W., 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. pp. 1187–1198 (2014)
19. Li, Y., Li, Q., Gao, J., Su, L., Zhao, B., Fan, W., Han, J.: On the discovery of evolving truth. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 675–684 (2015)
20. Li, Y., Li, Q., Gao, J., Su, L., Zhao, B., Fan, W., Han, J.: Conflicts to harmony: A framework for resolving conflicts in heterogeneous data by truth discovery. IEEE Transactions on Knowledge and Data Engineering **28**(8), 1986–1999 (2016)
21. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics. pp. 1273–1282. PMLR (2017)
22. Muñoz-González, L., Co, K.T., Lupu, E.C.: Byzantine-robust federated machine learning through adaptive model averaging. arXiv preprint arXiv:1909.05125 (2019)
23. Nguyen, T.D., Rieger, P., Chen, H., Yalame, H., Möllering, H., Fereidooni, H., Marchal, S., Miettinen, M., Mirhoseini, A., Zeitouni, S., et al.: Flame: Taming backdoors in federated learning. In: Proceedings of 31st USENIX Security Symposium. p. to appear (2022)
24. Ouyang, R.W., Kaplan, L.M., Toniolo, A., Srivastava, M., Norman, T.J.: Aggregating crowdsourced quantitative claims: Additive and multiplicative models. IEEE Transactions on Knowledge and Data Engineering **28**(7), 1621–1634 (2016)
25. Shen, S., Tople, S., Saxena, P.: Auror: Defending against poisoning attacks in collaborative deep learning systems. In: Proceedings of the 32nd Annual Conference on Computer Security Applications. pp. 508–519 (2016)
26. Tahmasebian, F., Lou, J., Xiong, L.: Robustfed: a truth inference approach for robust federated learning. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. pp. 1868–1877 (2022)
27. Wang, H., Sreenivasan, K., Rajput, S., Vishwakarma, H., Agarwal, S., Sohn, J.y., Lee, K., Papailiopoulos, D.: Attack of the tails: Yes, you really can backdoor federated learning. Advances in Neural Information Processing Systems **33**, 16070–16084 (2020)
28. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
29. Xie, C., Huang, K., Chen, P.Y., Li, B.: Dba: Distributed backdoor attacks against federated learning. In: International Conference on Learning Representations (2019)
30. Xu, C., Jia, Y., Zhu, L., Zhang, C., Jin, G., Sharif, K.: Tdfl: Truth discovery based byzantine robust federated learning. IEEE Transactions on Parallel and Distributed Systems **33**(12), 4835–4848 (2022)
31. Yin, D., Chen, Y., Kannan, R., Bartlett, P.: Byzantine-robust distributed learning: Towards optimal statistical rates. In: International Conference on Machine Learning. pp. 5650–5659. PMLR (2018)
32. Yin, X., Han, J., Philip, S.Y.: Truth discovery with multiple conflicting information providers on the web. IEEE Transactions on Knowledge and Data Engineering **20**(6), 796–808 (2008)