electronics

MDPI

Article

Byzantine Fault-Tolerant Federated Learning Based on Trustworthy Data and Historical Information

Xujiang Luo [1,2] and Bin Tang [1,2,*]

[1] Key Laboratory of Water Big Data Technology of Ministry of Water Resources, Hohai University, Nanjing 211100, China; luo_xj@hhu.edu.cn
[2] College of Computer Science and Software Engineering, Hohai University, Nanjing 211100, China
* Correspondence: cstb@hhu.edu.cn

Abstract: Federated learning (FL) is a highly promising collaborative machine learning method that preserves privacy by enabling model training on client nodes (e.g., mobile phones, Internet-of-Things devices) without sharing raw data. However, FL is vulnerable to Byzantine nodes, which can disrupt model performance, render training ineffective, or even manipulate the model by transmitting harmful gradients. In this paper, we propose a Byzantine fault-tolerant FL algorithm called federated learning with trustworthy data and historical information (FLTH). It utilizes a small trusted training dataset at the parameter server to filter out gradient updates from suspicious client nodes during model training, which provides both Byzantine resilience and convergence guarantee. It further introduces a historical information-based credibility assessment scheme such that the client nodes performing poorly over the long-term have a lower impact on the aggregation of gradients, thereby enhancing fault tolerance capability. Additionally, FLTH does not compromise the training efficiency of FL because of its low time complexity. Extensive simulation results show that FLTH achieves higher model accuracy compared to state-of-the-art methods under typical kinds of attack.

Keywords: federated learning; Byzantine fault tolerance; gradient aggregation

check for updates

Citation: Luo, X.; Tang, B. Byzantine Fault-Tolerant Federated Learning Based on Trustworthy Data and Historical Information. *Electronics* **2024**, *13*, 1540. https://doi.org/10.3390/electronics13081540

Academic Editor: Paulo Ferreira

Received: 25 March 2024
Revised: 13 April 2024
Accepted: 16 April 2024
Published: 18 April 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Federated learning (FL) is emerging as a widely adopted and distributed, privacy-preserving machine learning (ML) framework, wherein multiple client nodes cooperatively train an ML model with assistance from a parameter server [1,2]. With the proliferation of edge devices like mobile devices and Internet-of-Things (IoT) devices, FL enables efficient utilization of their local data and computational resources, thereby enhancing their capabilities [3,4]. The training process of FL consists of multiple rounds. In each round, the parameter server first broadcasts a global model to all client nodes. Then, each client node computes the gradient with the model on their local data, and sends it to the parameter server. Finally, the parameter server aggregates these gradients and updates the global model.

However, due to its distributed nature, some edge devices may be malicious; such malicious devices are commonly termed as Byzantine nodes. FL is vulnerable to Byzantine nodes, which can disrupt model performance, render training ineffective, or even manipulate the model by transmitting harmful gradients instead. For example, for the classic federated averaging (FedAvg) method [1] for gradient aggregation, it has been shown that the model can be arbitrarily manipulated even if there is only one Byzantine node but with an omniscient capability [5].

Many efforts have been devoted to design Byzantine fault-tolerant aggregation algorithms for FL. In Krum [5], Bulyan [6], RFA [7], and FLAME [8], the parameter server aggregate the gradients by finding a gradient that is close to majority gradients through clustering. In [9,10], the gradients are aggregated by combining common statistical methods

such as Median and Trimmed Mean. Moreover, refs. [11–15] address the evaluation and aggregation of client nodes over a historical perspective. They combine information from all previous rounds during every evaluation to counter Byzantine nodes. However, all these methods cannot guarantee effectiveness when more than a half of the client nodes are Byzantine, and most of them incur significant computational overhead.

Some recent work including Zeno [16,17], FLTrust [18], and Siren [19] has considered the introduction of trustworthy data. Among them, ref. [17] and FLTrust generate a reference gradient at parameter server and use the reference gradient to judge the accuracy of the gradients of client nodes, while Zeno and Siren directly evaluate gradients of client nodes on trustworthy data. These methods can be effective even if most of the client nodes are Byzantine. However, in all these methods, the evaluation of gradients of client nodes in each round is independent of the evaluations in other rounds. In other words, if the gradient of a client node passes the evaluation in the current round, it will be indiscriminately adopted for aggregation, no matter how the client node performs in the past rounds.

In this paper, we propose a fault-tolerant FL algorithm FLTH (federated learning with trustworthy data and historical information), which improves the existing trustworthy data-based methods by leveraging historical information to provide a more comprehensive assess of client nodes. Specifically, FLTH utilizes a small trusted training dataset at the parameter server to filter out gradient updates from suspicious client nodes during model training, which provides both Byzantine resilience and a convergence guarantee. It further introduces a historical information-based credibility assessment scheme such that the client nodes performing poorly over the long-term have a lower impact on the aggregation of gradients, thereby enhancing fault tolerance capability. The main contributions of this paper are as follows:

- We introduce FLTH, which provides Byzantine fault tolerance by assessing client nodes from a historical perspective based on trustworthy data.
- We show that FLTH has a low time complexity, which does not compromise the training efficiency, and establish its convergence result under mild assumptions.
- We conduct extensive simulations to evaluate the performance of FLTH. The simulation results show that FLTH achieves higher model accuracy compared to state-of-the-art methods under typical kinds of attack.

The reminder of the paper is organized as follows: In Section 2, we discuss the related work. In Section 3, we introduce the FL framework. In Section 4, we introduce FLTH and give the guarantee and convergence performance of our FLTH. In Section 5, we conduct the simulation and analyze the results. Finally, Section 6 concludes the whole paper.

2. Related Work

In this section, we first introduce some typical Byzantine attacks on FL, and then present a brief overview of existing Byzantine fault-tolerant algorithms for FL.

2.1. Attacks on Federated Learning

Due to the distributed feature of FL, each client node could potentially be malicious. These Byzantine nodes may adopt various attack methods to disrupt the model training process. With regard to the attacks that occur in the FL process, we mainly divide them into untargeted attacks and targeted attacks. The purpose of untargeted attacks is to undermine the accuracy of the model after aggregation, making the entire FL system unable to proceed normally or reducing the convergence speed of the model and the performance of the final converged model. The purpose of targeted attacks is not to undermine accuracy but to make the model produce specific errors in specific classifications, or to make the model output the results that the attacker wants after inputting data with specific features. Untargeted attacks mainly include Label-Flipping attack, Sign-Flipping attack, etc. Label-Flipping attack reverses the data labels during model training, for example, changing the labels of '0' to '9' in the MNIST dataset to '9' to '0', i.e., the label of '0' is '9', the label of '1' is '8', and

so on. Sign-Flipping attack reverses the sign of the gradient after training, for example, reversing the gradient of [1, −2, 3] to [−1, 2, −3]. Previous work [20–24] proposed various targeted attack methods. These attacks may deliberately steer the gradient purposefully in a certain direction or try to inject backdoor information into the model. For example, ref. [20] proposes a targeted attack with the concept of concealed and adversarial alternating minimization, which causes the model to misclassify on specific labels. Ref. [21] enables all Byzantine nodes to shift uniformly in a specific direction within a reasonable error range of the gradient.

2.2. Fault-Tolerant Algorithms

Clustering-Based Fault-Tolerant Approaches: The core of most fault-tolerant approaches lies in how to cluster and find results that are close to majority of the gradients. For example, methods like Krum [5], Bulyan [6], Median [9], RFA [7], etc., can find an appropriate aggregation result when the number of Byzantine nodes is less than $n/2$, where n is the number of client nodes, ensuring that the final result within a certain distance from the gradients of honest client nodes. However, these methods fail to tackle the scenario where the number of Byzantine nodes is more than $n/2$, and the time complexity of these algorithms is typically an order of magnitude higher than FedAvg.

Trustworthy Data-Based Fault-Tolerant Approaches: In fault-tolerant approaches based on trustworthy data, the key lies in how to utilize trustworthy data, which provides a fundamental guarantee for the aggregation. One method involves using a trusted dataset to directly evaluate the gradients returned by each client node. For example, Zeno [16] computes the loss value of the returned gradient under the trusted dataset and combines it with the gradient size to assign a score to each client node, thus assessing the trustworthiness of each client node. Based on this assessment, a certain number of gradients are selected for average aggregation according to their ranking. Similarly, Siren [19] employs the performance of the gradient under the trusted dataset to evaluate both the parameter server and the client node. The parameter server discards gradients with low accuracy, and the client node can also opt not to accept updates from the parameter server. Another method is to train a gradient in parallel using a trusted dataset on the parameter server. The returned gradients are indirectly evaluated based on the gradient trained by the trusted dataset. For example, ref. [17] trains the gradient on the parameter server, excludes those gradients whose distance exceeds the threshold from the gradient trained by the trusted dataset, and performs average aggregation on the remaining gradients. FLTrust [11] measures the cosine similarity between the returned gradient and the parameter server gradient and exclude some client nodes from aggregation. However, the existing approach using trustworthy data only measures the gradient in one round of training results and does not fully utilize the information brought by trustworthy data. Additionally, ref. [11] highlights issues arising from neglecting historical data usage.

History-Based Fault-Tolerant Approaches: The utilization of historical information introduces a historical perspective on fault tolerance within the federated training process. Various approaches exist for leveraging historical information. For example, ref. [13] regresses the gradient in the global model and employs the predicted values as a reference for the gradient. Moreover, refs. [11,12] utilize gradient momentum as an aggregation value to combat Byzantine nodes. However, these works have not significantly enhanced fault tolerance capabilities. Nevertheless, leveraging historical data represents a promising perspective for fault tolerance.

Unlike these existing works, in this paper we propose FLTH, which proposes a novel approach to leverage the advantages of both trustworthy data and historical information. Compared to the above clustering-based fault-tolerant approaches, FLTH can tolerate more than 50% Byzantine nodes and has a lower time complexity, while compared to the above trustworthy data based approaches, FLTH achieves a better fault tolerance capability by introducing a historical information-based credibility assessment scheme.

3. Problem Formulation

In this section, we first introduce the FL framework and then introduce both the attack model and the defender's ability.

3.1. Federated Learning Framework

As shown in Figure 1, we consider an FL system consisting of a parameter server and a set of n client nodes, and the client nodes communicate to the parameter server. The parameter server maintains a trusted dataset $\mathcal{D}_0$, and each client node i, $i = 1, \ldots, n$ possesses its own dataset $\mathcal{D}_i$, where each data point in $\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_n$ is drawn from an unknown distribution $\mathcal{X}$. The aim of this FL system is to collaboratively train a machine learning model $\mathbf{w} \in \mathbb{R}^d$ that minimizes the following expected loss function:

$$F(\mathbf{w}) = \mathbb{E}_{\mathcal{D} \sim \mathcal{X}}[f(\mathbf{w}, \mathcal{D})],$$

where $f(\mathbf{w}, \mathcal{D})$ denotes the loss function of the model $\mathbf{w}$ corresponding to dataset $\mathcal{D}$. Since the distribution $\mathcal{X}$ is unknown and the expectation is hard to evaluate exactly, the FL system usually seeks a model $\mathbf{w}$ that minimizes $f(\mathbf{w}, \mathcal{D})$ instead in practice, where $\mathcal{D} = \cup_{i=1}^{n} \mathcal{D}_i$.

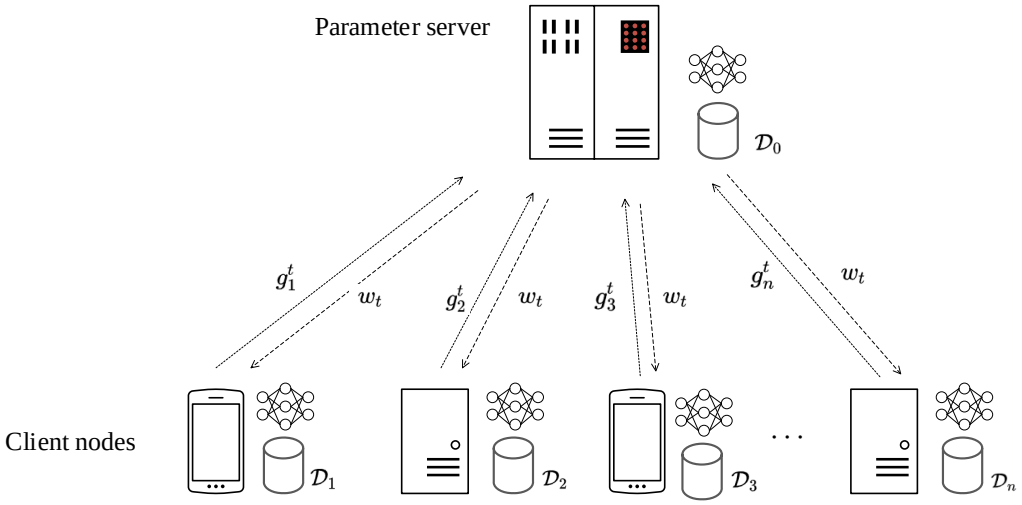


Figure 1. An illustration of FL framework.

The training process is executed in an iterative manner. More specifically, each t-th training round, $t = 1, 2, \ldots$, consists of the following three steps:

- First, the parameter server broadcasts the current global model $\mathbf{w}_{t-1}$ to every client node.
- If a client node is honest, it then computes its local model gradient $\mathbf{g}_i^t = \nabla f(\mathbf{w}_{t-1}, \mathcal{D}_i)$ and sends $\mathbf{g}_i^t$ to the parameter server. If otherwise, it sends an arbitrary vector $\mathbf{g}_i^t$ instead.
- After collecting every $\mathbf{g}_i^t$, $i = 1, \ldots, n$, the parameter server finally employs an aggregation algorithm **Aggr** to obtain an aggregated gradient update $\mathbf{Aggr}(\mathbf{g}_1^t, \ldots, \mathbf{g}_n^t)$, and subsequently utilizes the learning rate η to achieve a new global model $\mathbf{w}_t$ as follows:

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \mathbf{Aggr}(\mathbf{g}_1^t, \ldots, \mathbf{g}_n^t),$$

where the function **Aggr** plays a vital role.

3.2. Attack Model

Here, we primarily focus on scenarios where client nodes are malicious or manipulated by attackers. Malicious nodes may possess varying levels of capability, ranging from having knowledge limited to their local dataset and the globally transmitted model to being fully aware, meaning they have knowledge of all models and gradient information from all

client nodes but cannot directly influence the operations of the parameter server. Instead, they can only indirectly impact the model aggregation results of the parameter server by controlling the actions of client nodes. Additionally, malicious nodes can collude with each other to exert influence on the parameter server. We refer to these malicious client nodes as Byzantine nodes.

3.3. Defender's Ability

The parameter server has no permission to access the data of individual client nodes. It can only obtain gradient information transmitted by each client node. Moreover, it lacks information about which client nodes in the FL system might be malicious. However, similar to [16–19], we assume that the parameter server maintains a trusted dataset $\mathcal{D}_0$ that follows the distribution $\mathcal{X}$ as mentioned earlier, and has no intersection with the datasets of individual client nodes. Hence, the parameter server can also conduct model training using its own dataset $\mathcal{D}_0$ during each training round, which can be leveraged to defend against the Byzantine attack.

4. Federated Learning with Trustworthy Data and Historical Information

In this section, we introduce a novel algorithm named FLTH (federated learning with trustworthy data and historical information), which can mitigate the impact of Byzantine nodes. The convergence performance of FLTH is also provided.

4.1. Algorithm Description

In contrast to clustering-based fault tolerance methods, we introduce a novel algorithm known as FLTH. The core concept of FLTH is to establish and maintain a trusted dataset within the context of FL, aiming to enhance the trustworthiness of training and the quality of the model. We will evaluate in conjunction with the historical performance of the client nodes, so as to carry out comprehensive aggregation. In traditional FL, client nodes conduct model training solely based on their local data and then send the model gradient parameters to the parameter server for global model updates. However, in FLTH, we introduce additional data to ensure the security and effectiveness of FL.

Next, we will specifically introduce our FLTH algorithm. Our method is grounded in the following core principles:

Introducing Trustworthy Reference Gradients: In each t-th training round, the parameter server utilizes its computational resources to compute a gradient $\mathbf{g}_0^t = \nabla f(\mathbf{w}_{t-1}, \mathcal{D}_0)$ using its own trusted dataset $\mathcal{D}_0$. This gradient $\mathbf{g}_0^t$ represents reliable gradient information, serving as a reference point for evaluating the credibility of the gradients of client nodes. Henceforth, in the following, we will refer to $\mathbf{g}_0^t$ as the reference gradient.

Filtering: The gradient of each honest client node usually exhibits certain similarity with the reference gradient provided by the parameter server in a same training round. This implies that, if the gradient of some client node is quite dissimilar with the reference gradient $\mathbf{g}_0^t$, the client node is suspected to be Byzantine. In order to avoid the potential influence of gradients sent by possible Byzantine nodes on the aggregation, we exclude the gradients that are quite dissimilar with the reference gradient from the aggregation process. More specifically, we use the widely adopted Euclidean distance to measure the similarity between gradients. If the gradient of client node i satisfies

$$\|\mathbf{g}_i^t - \mathbf{g}_0^t\| > k\|\mathbf{g}_0^t\|,$$

where k is a fixed constant referred to as filtering parameter, then client node i will be excluded from the aggregation process in this training round. We denote the set of the client nodes that are not excluded by $\mathcal{S}_t$.

Inverse Distance Weighting: In order to evaluate the impact of the gradient $\mathbf{g}_i^t$ of each client node i on the aggregation in the t-th round, we first introduce a credibility value $\bar{r}_i^t$ for each client node i. Generally speaking, if a gradient exhibits higher similarity with the reference gradient, a higher credibility value should be assigned. To guarantee this property,

we employ an inverse distance weighting approach [25]. Specifically, if $i \notin \mathcal{S}_t$, then we set $\tilde{r}_i^t = 0$. Otherwise, the credibility value $\tilde{r}_i^t$ is set as the p-th power of the reciprocal of the Euclidean distance, i.e.,

$$\tilde{r}_i^t = \left(\frac{1}{\|\mathbf{g}_i^t - \mathbf{g}_0^t\|} \right)^p,$$

where p is a constant parameter used for adjusting the sensitivity of the credibility value in this round to the distance. The larger p is, the greater the role of gradient with higher similarity will play, further limiting the malicious ability of Byzantine nodes, but it will also affect the role of gradients with lower similarity but coming from honest client nodes in the aggregation. Therefore, by adjusting the power exponent p, we can achieve different levels of fault tolerance.

Normalization of Node Trustworthiness: To eliminate the discrepancy in the credibility values between different rounds and facilitate the utilization of historical information later, we normalize the credibility value $\tilde{r}_i^t$ of each client node i to a value r_i^t in the interval $[0, 1]$, given by

$$r_i^t = \frac{\tilde{r}_i^t}{\sum_{j=1}^n \tilde{r}_j^t}.$$

Historical Perspective: One straightforward aggregation method is to use $\sum_{i \in \mathcal{S}_t} r_i^t \mathbf{g}_i^t$ as the aggregation result, which can provide a certain fault-tolerant capability. However, the normalized credibility value of a single round cannot fully reflect the long-term performance of client nodes, making the aggregation method less useful to distinguish Byzantine nodes from honest client nodes. Hence, the fault-tolerant capability of this method is limited. To overcome this drawback, our idea is to penalize client nodes with poor historical performance so their influence on the aggregation result is restricted. For this purpose, we introduce the utilization of the historical information of credibility values of client nodes in each round to evaluate each client node from the long-term perspective. More specifically, we leverage the common exponential moving average $\bar{r}_i^t$ of $r_i^1, r_i^2, \ldots, r_i^t$, referred to as historical credibility value, i.e.,

$$\bar{r}_i^t = \beta \bar{r}_i^{t-1} + (1 - \beta) r_i^t,$$

where $\bar{r}_i^0 = 0$, and β is a parameter that balances the historical credibility value and the normalized credibility value of the current round.

Weighted Average Aggregation of Model: Now we can have the following basic aggregation method

$$\mathbf{Aggr}_0(\mathbf{g}_1^t, \ldots, \mathbf{g}_n^t) = \sum_{i \in \mathcal{S}_t} \frac{\bar{r}_i^t}{R_t} \mathbf{g}_i^t,$$

where $R_t = \sum_{i \in \mathcal{S}_t} \bar{r}_i^t$ is a parameter such that the sum of coefficients of $\mathbf{g}_i^t$ is equal to 1.

Note that the reference gradient $\mathbf{g}_0^t$ is ignored by the above aggregation method, which is wasteful. To integrate $\mathbf{g}_0^t$, we introduce the following improved aggregation method:

$$\mathbf{Aggr}(\mathbf{g}_1^t, \ldots, \mathbf{g}_n^t) = \frac{1}{|\mathcal{S}_t| + 1} \mathbf{g}_0^t + \frac{|\mathcal{S}_t|}{|\mathcal{S}_t| + 1} \mathbf{Aggr}_0(\mathbf{g}_1^t, \ldots, \mathbf{g}_n^t),$$

where the weight of $\mathbf{g}_0^t$ is set to be inversely proportional to the number of participating nodes including the parameter server.

The following result shows that FLTH is computationally efficient. In particular, it has the same time complexity as the well-known FedAvg aggregation algorithm, which does not provide Byzantine fault tolerance.

Proposition 1. *The time complexity of our FLTH algorithm is $O(nd)$.*

Proof. Clearly, the time cost of calculating the credibility value of each client node is $O(d)$. Hence, obtaining the historical credibility values of all n client nodes requires $O(nd)$ time. Furthermore, the weighted average aggregation requires $O(nd)$ time. Therefore, the whole time cost of FLTH is $O(nd)$. □

The whole pseudocode of FLTH algorithm is given in Algorithm 1.

Algorithm 1 FLTH Algorithm

Input:
client nodes number n, initial model $\mathbf{w}_0$, learning rate η
momentum parameter β, filter parameter k
training round T, fault tolerance parameter p
Initialize:
client nodes global credibility value $\bar{r}_i^0 = 0 (i = 1, \ldots, n)$
Output:
Global model $\mathbf{w}_T$
for $t = 1$ **to** T **do**
parameter server send model $\mathbf{w}_{t-1}$ to all client nodes

Client node:
for $i = 1$ **to** n **do**
client node i calculates $\mathbf{g}_i^t = \nabla F(\mathbf{w}_{t-1}, \mathcal{D}_i)$
send $\mathbf{g}_i^t$ to parameter server
end for

Parameter Server:
parameter server calculates $\mathbf{g}_0^t = \nabla F(\mathbf{w}_{t-1}, \mathcal{D}_0)$
let $\mathcal{S}_t = \{\}$
for $i = 1$ **to** n **do**
if $\|\mathbf{g}_i^t - \mathbf{g}_0^t\| \leq k\|\mathbf{g}_0^t\|$ **then**
calculate client node i's credibility value $r_i^t = (\frac{1}{\|\mathbf{g}_i^t - \mathbf{g}_0^t\|})^p$
$\mathcal{S}_t = \mathcal{S}_t \cup \{i\}$
else
client node i's credibility value $\tilde{r}_i^t = 0$
end if
end for
for $i = 1$ **to** n **do**
client node credibility value normalization $r_i^t = \frac{\tilde{r}_i^t}{\sum_{j=1}^n \tilde{r}_j^t}$
end for
for $i = 1$ **to** n **do**
calculate client node i's global credibility value $\bar{r}_i^t = \beta \bar{r}_i^{t-1} + (1 - \beta) r_i^t$
end for
calculate sum of global credibility value $R_t = \sum_{i \in \mathcal{S}_t} \bar{r}_i^t$
$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta(\frac{1}{|\mathcal{S}_t|+1}\mathbf{g}_0^t + \frac{|\mathcal{S}_t|}{|\mathcal{S}_t|+1} \sum_{i \in \mathcal{S}_t} \frac{\bar{r}_i^t}{R_t} \mathbf{g}_i^t)$

end for
return $\mathbf{w}_T$

4.2. Convergence Result

For the convergence analysis of our proposed FLTH algorithm, we start with some assumptions that have been also used in [17,18].

Assumption 1. *The expected loss function F is M-strongly convex and its gradient is L-Lipschitz in the space Θ, so we have:*

$$F(\mathbf{w}') \geq F(\mathbf{w}) + \langle \nabla F(\mathbf{w}), \mathbf{w}' - \mathbf{w} \rangle + \frac{M}{2}\|\mathbf{w}' - \mathbf{w}\|^2$$
$$\|\nabla F(\mathbf{w}') - \nabla F(\mathbf{w})\| \leq L\|\mathbf{w}' - \mathbf{w}\|$$

where $\mathbf{w}, \mathbf{w}'$ denotes any vector in Θ, $\|\cdot\|$ denotes l_2 norm, and $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors.

Assumption 2. *There exist positive constants σ_1 and α_1 such that for any unit vector $\mathbf{v}$, $\langle \nabla f(\mathbf{w}, \mathcal{D}), \mathbf{v} \rangle$ is sub-exponential with σ_1 and α_1, i.e.,*

$$\sup \mathbb{E}[\exp(\lambda \langle \nabla f(\mathbf{w}, \mathcal{D}), \mathbf{v} \rangle)] \leq \exp(\frac{\sigma_1^2 \lambda^2}{2}) \qquad \forall |\lambda| \leq \frac{1}{\alpha_1}$$

Assumption 3. *The gradient difference $h(\mathbf{w}, \mathcal{D}) \triangleq \nabla f(\mathbf{w}, \mathcal{D}) - \nabla f(\mathbf{w}^*, \mathcal{D})$ for any $\mathbf{w} \in \Theta$ is bounded. Formally, there exist positive constants σ_2 and α_2 such that for any $\mathbf{w} \in \Theta$ with $\mathbf{w} \neq \mathbf{w}^*$ and any unit vector $\mathbf{v}$, we have:*

$$\sup \mathbb{E}[\exp(\frac{\lambda \langle h(\mathbf{w}, \mathcal{D}) - \mathbb{E}[h(\mathbf{w}, \mathcal{D})], \mathbf{v} \rangle}{\|\mathbf{w} - \mathbf{w}^*\|})] \leq \exp(\frac{\sigma_2^2 \lambda^2}{2}) \qquad \forall |\lambda| \leq \frac{1}{\alpha_2}$$

Assumption 4. *The empirical loss function $f(\mathbf{w}, \mathcal{D})$ is L'-Lipschitz with high probability, i.e., for any $\delta \in (0, 1)$, there exists $L' = L'(\delta)$ such that:*

$$\Pr\left\{ \sup_{\mathbf{w}, \mathbf{w}' \in \Theta, \mathbf{w} \neq \mathbf{w}'} \|\nabla f(\mathbf{w}, \mathcal{D}) - \nabla f(\mathbf{w}', \mathcal{D})\| \leq L'\|\mathbf{w} - \mathbf{w}'\| \right\} \geq 1 - \frac{\delta}{3}$$

Assumption 5. *The root dataset $\mathcal{D}_0$ and each client node dataset $\mathcal{D}_i (i = 1, \ldots, n)$ are independently sampled from distribution $\mathcal{X}$.*

The convergence result of the FLTH algorithm is given in the following theorem.

Theorem 1. *Under Assumptions 1–5 and $\Theta \subset \{\boldsymbol{\theta} : \|\boldsymbol{\theta} - \mathbf{w}^*\|\} \leq r\sqrt{d}$ for some $r > 0$, setting learning rate $\eta = \frac{M}{L^2}$, it is guaranteed with at least a probability of $1 - \delta$ that:*

$$\|\mathbf{w}_t - \mathbf{w}^*\| \leq (1 - \rho)^t \|\mathbf{w}_0 - \mathbf{w}^*\| + 4\eta\Delta_1(1 + k)/\rho,$$

where

$$\begin{aligned}
\rho &= 1 - \sqrt{1 - M^2/L^2} + 8\eta\Delta_2(1 + k) + \eta k L \\
\Delta_1 &= \sqrt{2}\sigma_1 \sqrt{\frac{d \log 6 + \log(\frac{3}{\delta})}{|\mathcal{D}_0|}} \\
\Delta_2 &= \sqrt{2}\sigma_2 \sqrt{\frac{\gamma_1 + \gamma_2}{|\mathcal{D}_0|}} \\
\gamma_1 &= d \log 18 + d \log(\frac{\max\{L, L'\}}{\sigma_2}) \\
\gamma_2 &= 0.5d \log\left(\frac{|\mathcal{D}_0|}{d}\right) + \log\left(\frac{3}{\delta}\right) + \log\left(\frac{2r\sigma_2^2\sqrt{|\mathcal{D}_0|}}{\alpha_2\sigma_1}\right)
\end{aligned}$$

and d is the dimension of the model.

Proof. This convergence result of our FLTH algorithm is based on the convergence result of the methods proposed in [17,18]. In the method proposed in [17,18], the parameter server also maintains a trusted dataset, trains reference gradients based on the dataset, and also excludes the received gradients of client nodes far from the reference gradient in terms of Euclidean distance from aggregation, which are the same methods as our FLTH. The only difference between the method and our FLTH is that the weights of aggregated gradients are different. Specifically, FLTH leverages the historical information to obtain weights while the former does not. However, we observe that the convergence result presented in [17,18] is unrelated to the weights in essence. So, we can obtain the convergence result of FLTH by following the same lines in [17,18]. The details are omitted here. □

5. Performance Evaluation

In this section, we evaluate the performance of our proposed FLTH algorithm under several typical types of attacks.

5.1. Experiment Setup

We consider a scenario consisting of a parameter server and 20 client nodes. We conduct experiments on the MNIST [26] and CIFAR-10 [27] datasets. The MNIST dataset contains 60,000 gray-scale images for training and 10,000 gray-scale images for testing. The CIFAR-10 dataset contains 50,000 color images for training and 10,000 color images for testing. All client nodes and parameter servers will distribute data from the dataset in an average and i.i.d. manner. For MNIST, we train a convolutional neural network (CNN) with a learning rate $\eta = 0.01$. For CIFAR-10 we train ResNet [28] with a learning rate $\eta = 0.05$.

5.2. Attacks

Similar to [11], we consider different kinds of attacks including both untargeted attack and targeted attack. Two typical untargeted attacks are considered as follows:

- **Sign-Flipping attack**: where each parameter of the gradient sent to the parameter server by a Byzantine node is set to be the opposite value of the training gradient.
- **Label-Flipping attack**: For Label-Flipping, since both the MNIST and CIFAR-10 datasets have only the labels '0' to '9', we will modify their labels to range from '9' to '0', respectively.

The targeted attack is considered as follows:

- **ALIE attack** [21]: ALIE is a targeted attack method. It allows all colluding client nodes to send the same value, causing errors in the final result. In this paper, we adopt the default settings in [21] as our chosen attack strategy, where all the Byzantine nodes send $Mean(\{\mathbf{g}_i^t, i \in \mathcal{H}\}) - z^{max} \cdot Std(\{\mathbf{g}_i^t, i \in \mathcal{H}\})$. Here, $\mathcal{H}$ is the set of honest client nodes, $Std()$ is standard deviation function, and $z^{max} = \arg\max_z \phi(z) < \frac{\lceil \frac{n}{2} \rceil - 1}{n-f}$, where $\phi()$ denotes cumulative standard normal function, and f denotes the number of Byzantine nodes.

5.3. Fault Tolerance Algorithms for Comparison

We compare our FLTH with FedAvg [1] and four state-of-the-art fault tolerance algorithms: Krum [5], Median [9], FLTrust [18], and CC + Momentum [11], which use different **Aggr** functions. The details are given as follows.

Krum [5]: Krum introduces a scoring mechanism. Considering the case where m Byzantine nodes are present, it calculates the sum of squared Euclidean distances to the

nearest $n-m-2$ client nodes for each client node. Subsequently, the gradient of the client node with the lowest score is selected as the aggregation result. It can be expressed as:

$$s = \arg\min_i \sum_{j \in \mathcal{N}_i} \|\mathbf{g}_i - \mathbf{g}_j\|_2^2$$
$$Krum(\mathbf{g}_1, \ldots, \mathbf{g}_n) = \mathbf{g}_s$$

where $\mathcal{N}_i$ denotes the set of the $n-m-2$ nearest client nodes to client node i. In our experiments, we assume that the parameter server knows the exact number m of Byzantine nodes.

Median [9]: Its **Aggr** function calculates the median value of each parameter in all client nodes' gradients as the corresponding parameter value in the aggregation results.

FLTrust [18]: FLTrust computes a reference gradient $\mathbf{g}_0$ based on the trusted dataset at the parameter server, measuring the cosine similarity between transmitted gradients and parameter server gradients to obtain a trust score, excluding gradients with negative cosine similarity, and reweighting gradients based on the trust score. It can be expressed as:

$$TS_i = max(0, \frac{\langle \mathbf{g}_i, \mathbf{g}_0 \rangle}{\|\mathbf{g}_i\|_2 \cdot \|\mathbf{g}_0\|_2})$$
$$FLTrust(\mathbf{g}_1, \ldots, \mathbf{g}_n) = \sum_{i=1}^{n} \frac{TS_i \cdot \frac{\mathbf{g}_i \cdot \|\mathbf{g}_0\|_2}{\|\mathbf{g}_i\|_2}}{\sum_{j=1}^{n} TS_j}$$

CC + Momentum [11]: CC + Momentum consists of two parts: CC (centered clipping) and Momentum. CC is implemented at the parameter server, while Momentum is implemented at each honest client node. Honest client nodes send their gradients with momentum instead of the original gradients. It can be expressed as:

$$\mathbf{m}_i^t = (1-\beta)\mathbf{g}_i^{t-1} + \beta \mathbf{m}_i^{t-1}$$
$$\mathbf{w}_t = CC(\mathbf{m}_1^t, \ldots, \mathbf{m}_n^t) = \mathbf{w}_{t-1} + \frac{1}{n}\sum_{i=1}^{n}(\mathbf{m}_i^t - \mathbf{w}_{t-1}) min(1, \frac{\tau}{\|\mathbf{m}_i^t - \mathbf{w}_{t-1}\|_2})$$

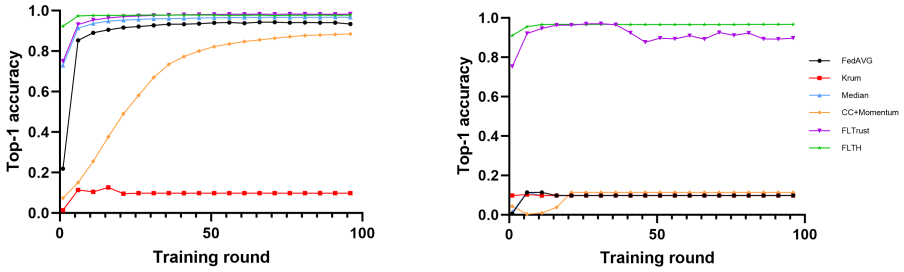
where all parameters in $\mathbf{m}_i^0 (i = 1, 2, \ldots, n)$ are 0. β is the momentum parameter set by parameter server. τ is the radius hyperparameter set by parameter server, which is set as 0.1 in our experiments.

The fault-tolerant methods including Krum [5], Median [9], and CC + Momentum [11] can only tolerate less than 50% of Byzantine nodes. On the other hand, the method of FLTrust can tolerate over 50% and even a higher proportion as demonstrated in [18]. In order to compare with these methods fairly, we consider two proportions of Byzantine nodes in our experiments, namely 40% and 80%.

Furthermore, in FLTH, we set $\beta = 0.5$ and $k = 1$ to achieve a balance between robustness and the retention of valuable information from honest client nodes. The default value of parameter p is set to be 2.

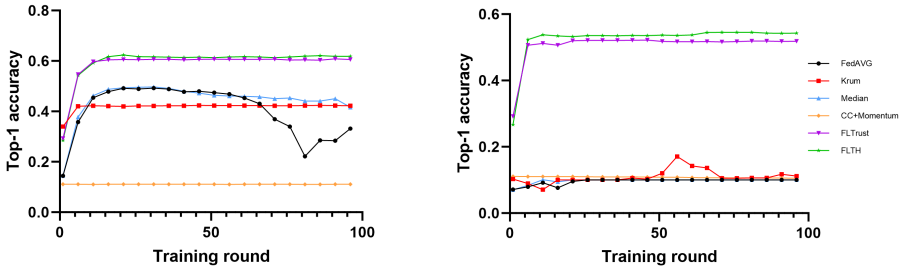
5.4. Experimental Results

The experimental results in Table 1 and Figures 2–7 show the performance of various algorithms under different attacks and varying proportions of Byzantine nodes. We utilize top-1 accuracy on the test sets as the evaluation metric. Subsequently, we will analyze our results across various attacks.



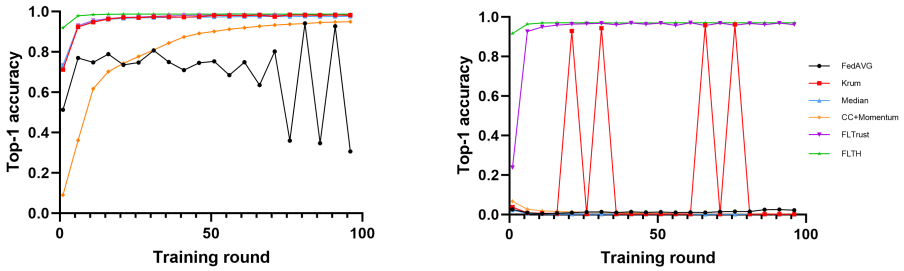
(**a**) 40% Sign-Flipping (**b**) 80% Sign-Flipping

Figure 2. MNIST dataset in 20 clients with Sign-Flipping attack.



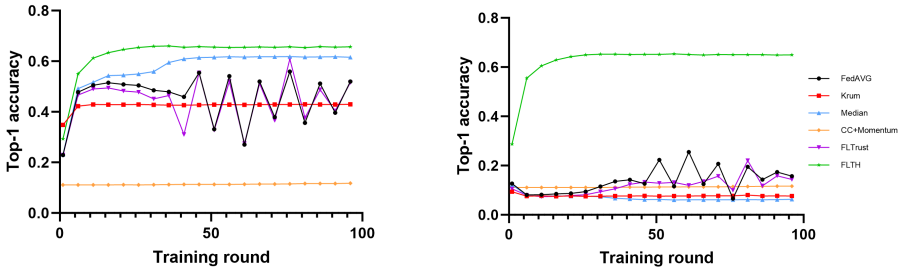
(**a**) 40% Sign-Flipping (**b**) 80% Sign-Flipping

Figure 3. CIFAR-10 dataset in 20 clients with Sign-Flipping attack.



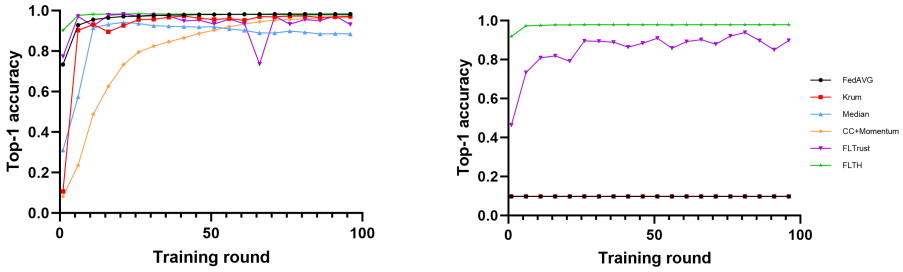
(**a**) 40% Label-Flipping (**b**) 80% Label-Flipping

Figure 4. MNIST dataset in 20 clients with Label-Flipping attack.



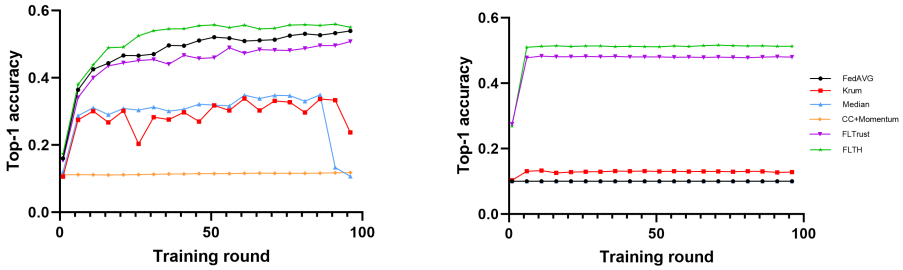
(**a**) 40% Label-Flipping (**b**) 80% Label-Flipping

Figure 5. CIFAR-10 dataset in 20 clients with Label-Flipping attack.



(**a**) 40% ALIE attack (**b**) 80% ALIE attack

Figure 6. MNIST dataset in 20 clients with ALIE attack.



(**a**) 40% ALIE attack (**b**) 80% ALIE attack

Figure 7. CIFAR-10 dataset in 20 clients with ALIE attack.

Table 1. Top-1 accuracy comparison of different aggregation methods under different circumstances.

			FedAvg	Krum	Median	CC+ Momentum	FLTrust	FLTH
40% Byzantine nodes	MNIST	Sign-Flipping	0.9077	0.0980	0.9665	0.8880	**0.9824**	0.9775
		Label-Flipping	0.2963	0.9830	0.9757	0.9505	0.9800	**0.9870**
		ALIE	**0.9841**	0.9735	0.8771	0.9688	0.9279	0.9779
	CIFAR-10	Sign-Flipping	0.2791	0.4228	0.4319	0.1108	0.6046	**0.6170**
		Label-Flipping	0.6070	0.4296	0.6159	0.1178	0.6113	**0.6558**
		ALIE	0.5423	0.2515	0.1000	0.1180	0.5045	**0.5652**
80% Byzantine nodes	MNIST	Sign-Flipping	0.0980	0.0980	0.0980	0.1135	0.8956	**0.9669**
		Label-Flipping	0.0182	0.0032	0.0014	0.0023	0.9602	**0.9704**
		ALIE	0.0980	0.0980	0.0980	0.0980	0.9029	**0.9787**
	CIFAR-10	Sign-Flipping	0.1000	0.1092	0.1000	0.1049	0.5187	**0.5428**
		Label-Flipping	0.1623	0.0773	0.0619	0.1167	0.1293	**0.6485**
		ALIE	0.1000	0.1283	0.1000	0.1000	0.4804	**0.5132**

Results on Sign-Flipping attack: The performances of the algorithms on the MNIST and CIFAR-10 datasets under the Sign-Flipping attack with 40% and 80% Byzantine nodes are shown in Figure 2 and Figure 3, respectively.

- In the MNIST dataset, under the condition of 40% Byzantine nodes, except for Krum, all algorithms exhibit high accuracy. Among them, FLTH, FLTrust, and Median perform well. Additionally, FedAvg also demonstrates fault tolerance capabilities. We believe this is because Byzantine nodes only send gradients opposite to the original gradients, allowing a large proportion of honest client nodes to partially offset the influence of these Byzantine nodes. Although CC + Momentum is fault-tolerant, it

lags behind most methods to some extent. Under the condition of 80% Byzantine nodes, only FLTH and FLTrust demonstrate fault tolerance capabilities, with FLTH being approximately 6% higher in accuracy compared to FLTrust.

- In the CIFAR-10 dataset, under the condition of 40% Byzantine nodes, only FLTH and FLTrust perform well, while Median and Krum are significantly disrupted by Byzantine nodes. Under the condition of 80% Byzantine nodes, only FLTH and FLTrust demonstrate fault tolerance capabilities, with FLTH being approximately 2.5% higher in accuracy compared to FLTrust.

Results on Label-Flipping attack: The performances of the algorithms on the MNIST and CIFAR-10 datasets under the Label-Flipping attack with 40% and 80% Byzantine nodes are shown in Figure 4 and Figure 5, respectively.

- In the MNIST dataset, all methods except FedAvg exhibit fault tolerance with 40% Byzantine nodes, showing little difference in performance. The reason for this is that the gradients of the Byzantine nodes differ significantly from those of honest client nodes' gradients. Under the condition of 80% Byzantine nodes, only FLTH and FLTrust demonstrate fault tolerance capabilities, with no significant difference in accuracy performance.
- In the CIFAR-10 dataset, under the condition of 40% Byzantine nodes, FLTH and Median perform well, while other fault-tolerant methods are affected on different levels. Under the condition of 80% Byzantine nodes, only FLTH achieves fault tolerance.

Results on ALIE attack: The performances of the algorithms on the MNIST and CIFAR-10 datasets under the ALIE attack with 40% and 80% Byzantine nodes are shown in Figure 6 and Figure 7, respectively.

- In the MNIST dataset, under the condition of 40% Byzantine nodes, all methods are minimally affected in terms of accuracy, with Median being the most affected. We attribute this to the ALIE attack, which causes smaller but more definite gradient shifts. Under the condition of 80% Byzantine nodes, FLTH and FLTrust demonstrate fault tolerance, with FLTH achieving approximately 8% higher accuracy compared to FLTrust. We believe FLTrust is impacted significantly because it evaluates nodes in each round of aggregation independently.
- In the CIFAR-10 dataset, under the condition of 40% Byzantine nodes, FLTH, FedAvg, and FLTrust perform well, while other methods are significantly impacted. Under the condition of 80% Byzantine nodes, only FLTH and FLTrust exhibit fault tolerance, with FLTH achieving approximately 3% higher accuracy compared to FLTrust.

Effect of parameter p in FLTH: Here, we set p with three different values of 1, 2, and 4. The performances of FLTH on the MNIST and CIFAR-10 datasets under three types of attack with 40% and 80% Byzantine nodes and different values of parameter p are shown in Table 2.

- For the MNIST dataset, FLTH with $p = 2$ performs slightly better than that with $p = 1$ and $p = 4$ in all circumstances.
- For the CIFAR-10 dataset, FLTH with $p = 2$ performs slightly better than that with $p = 1$ and $p = 4$ in most cases. One exception is that FLTH with $p = 1$ achieves a low accuracy when there are 80% Byzantine nodes performing the Label-Flipping attack, where $p = 1$ cannot provide sufficient fault tolerance capability.

These results suggest that $p = 2$ is a satisfactory choice.

In summary, our method FLTH performs best among these methods under all three kinds of attacks, and can achieve a high model accuracy even when 80% of the client nodes are Byzantine. Additionally, the accuracy performance of FLTH oscillated less during the training process compared to other methods, leading to a smoother convergence.

Table 2. Top-1 accuracy comparison of FLTH under different parameter p.

			FLTH, $p = 1$	**FLTH, $p = 2$**	**FLTH, $p = 4$**
40% Byzantine nodes	MNIST	Sign-Flipping	0.9702	**0.9775**	0.9705
		Label-Flipping	0.9696	**0.9870**	0.9706
		ALIE	0.9709	**0.9779**	0.9699
	CIFAR-10	Sign-Flipping	**0.6230**	0.6170	0.6174
		Label-Flipping	0.5291	**0.6558**	0.6205
		ALIE	0.5477	0.5652	**0.5936**
80% Byzantine nodes	MNIST	Sign-Flipping	0.9632	**0.9669**	0.9643
		Label-Flipping	0.9701	**0.9704**	**0.9704**
		ALIE	0.9700	**0.9787**	0.9698
	CIFAR-10	Sign-Flipping	**0.5443**	0.5428	0.5303
		Label-Flipping	0.2058	**0.6485**	0.5927
		ALIE	0.5131	0.5132	**0.5142**

6. Concluding Remarks

In this paper, we introduce FLTH, a fault-tolerant FL method, by providing a comprehensive assessment of client nodes based on trustworthy data and historical information. FLTH exhibits fault tolerance even in the presence of a high proportion of malicious nodes, and has a low computation cost. Simulation results show that FLTH achieves better performance in terms of both model accuracy and training stability compared to state-of-the-art methods.

Recall that we assumed the datasets of clients as well as the trusted dataset maintained by the parameter server follow the same unknown data distribution. When the datasets are not independently and identically distributed, especially when the data collected by the parameter server are over specialized, our algorithm FLTH may struggle to accurately reflect the trustworthiness of different client nodes. This will be further investigated in the future. Additionally, we would also like to further optimize the credibility assessment algorithm, enhance anomaly detection strategies, and explore the historical behavior of client nodes during multi-round training processes in greater depth.

Author Contributions: Conceptualization, B.T.; Methodology, X.L.; Validation, X.L.; Writing—review & editing, X.L. and B.T. All authors have read and agreed to the published version of the manuscript.

Funding: This paper is supported by the National Key R&D Program of China under Grant No. 2023YFC3006505, the National Natural Science Foundation of China under Grant No. 61872171, and the Future Network Scientific Research Fund Project under Grant No. FNSRFP-2021-ZD-07.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
2. Sheller, M.J.; Edwards, B.; Reina, G.A.; Martin, J.; Pati, S.; Kotrotsou, A.; Milchenko, M.; Xu, W.; Marcus, D.; Colen, R.R.; et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Sci. Rep.* **2020**, *10*, 12598. [CrossRef] [PubMed]
3. Hard, A.; Rao, K.; Mathews, R.; Ramaswamy, S.; Beaufays, F.; Augenstein, S.; Eichner, H.; Kiddon, C.; Ramage, D. Federated learning for mobile keyboard prediction. *arXiv* **2018**, arXiv:1811.03604.

4. Chen, M.; Yang, Z.; Saad, W.; Yin, C.; Poor, H.V.; Cui, S. A joint learning and communications framework for federated learning over wireless networks. *IEEE Trans. Wirel. Commun.* **2020**, *20*, 269–283. [CrossRef]
5. Blanchard, P.; El Mhamdi, E.M.; Guerraoui, R.; Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. In Proceedings of the Advances in Neural Information Processing Systems 30 (NeurIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
6. Guerraoui, R.; Rouault, S. The hidden vulnerability of distributed learning in byzantium. In Proceedings of the International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2018; pp. 3521–3530.
7. Pillutla, K.; Kakade, S.M.; Harchaoui, Z. Robust aggregation for federated learning. *IEEE Trans. Signal Process.* **2022**, *70*, 1142–1154. [CrossRef]
8. Nguyen, T.D.; Rieger, P.; De Viti, R.; Chen, H.; Brandenburg, B.B.; Yalame, H.; Möllering, H.; Fereidooni, H.; Marchal, S.; Miettinen, M.; et al. FLAME: Taming backdoors in federated learning. In Proceedings of the 31st USENIX Security Symposium, Boston, MA, USA, 10–12 August 2022; pp. 1415–1432.
9. Yin, D.; Chen, Y.; Kannan, R.; Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 5650–5659.
10. Chen, Y.; Su, L.; Xu, J. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proc. ACM Meas. Anal. Comput. Syst.* **2017**, *1*, 1–25. [CrossRef]
11. Karimireddy, S.P.; He, L.; Jaggi, M. Learning from history for byzantine robust optimization. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 5311–5319.
12. Farhadkhani, S.; Guerraoui, R.; Gupta, N.; Pinot, R.; Stephan, J. Byzantine machine learning made easy by resilient averaging of momentums. In Proceedings of the International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022; pp. 6246–6283.
13. Fu, S.; Xie, C.; Li, B.; Chen, Q. Attack-resistant federated learning with residual-based reweighting. *arXiv* **2019**, arXiv:1912.11464.
14. Alistarh, D.; Allen-Zhu, Z.; Li, J. Byzantine stochastic gradient descent. In Proceedings of the Advances in Neural Information Processing Systems 31 (NeurIPS 2018), Montréal, QC, Canada, 3–8 December 2018.
15. Li, Z.; Liu, L.; Zhang, J.; Liu, J. Byzantine-robust federated learning through spatial-temporal analysis of local model updates. In Proceedings of the 2021 IEEE 27th International Conference on Parallel and Distributed Systems, Beijing, China, 14–16 December 2021; pp. 372–379.
16. Xie, C.; Koyejo, S.; Gupta, I. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 6893–6901.
17. Cao, X.; Lai, L. Distributed gradient descent algorithm robust to an arbitrary number of byzantine attackers. *IEEE Trans. Signal Process.* **2019**, *67*, 5850–5864. [CrossRef]
18. Cao, X.; Fang, M.; Liu, J.; Gong, N. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. In Proceedings of the Network and Distributed System Security Symposium, Virtual, 21–25 February 2021.
19. Guo, H.; Wang, H.; Song, T.; Hua, Y.; Lv, Z.; Jin, X.; Xue, Z.; Ma, R.; Guan, H. Siren: Byzantine-robust federated learning via proactive alarming. In Proceedings of the ACM Symposium on Cloud Computing, Seattle, WA, USA, 1–4 November 2021; pp. 47–60.
20. Bhagoji, A.N.; Chakraborty, S.; Mittal, P.; Calo, S. Analyzing federated learning through an adversarial lens. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 634–643.
21. Baruch, G.; Baruch, M.; Goldberg, Y. A little is enough: Circumventing defenses for distributed learning. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019.
22. Xie, C.; Koyejo, O.; Gupta, I. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In Proceedings of the Uncertainty in Artificial Intelligence, Online, 3–6 August 2020; pp. 261–270.
23. Fang, M.; Cao, X.; Jia, J.; Gong, N. Local model poisoning attacks to Byzantine-Robust federated learning. In Proceedings of the 29th USENIX Security Symposium, Boston, MA, USA, 12–14 August 2020; pp. 1605–1622.
24. Xie, C.; Huang, K.; Chen, P.Y.; Li, B. Dba: Distributed backdoor attacks against federated learning. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
25. Shepard, D. A two-dimensional interpolation function for irregularly-spaced data. In Proceedings of the 23rd ACM National Conference, New York, NY, USA, 27–29 August 1968; pp. 517–524.
26. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
27. Krizhevsky, A.; Hinton, G. Learning Multiple Layers of Features from Tiny Images. Technical Report. 2009. Available online: www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf (accessed on 8 January 2024).
28. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.