electronics

MDPI

Article

WGM-dSAGA: Federated Learning Strategies with Byzantine Robustness Based on Weighted Geometric Median

Xiaoxue Wang, Hongqi Zhang, Anas Bilal, Haixia Long * and Xiaowen Liu *

College of Information Science Technology, Hainan Normal University, Haikou 571158, China
* Correspondence: myresearch_hainnu@163.com (H.L.); userlxw@126.com (X.L.)

Abstract: Federated learning techniques accomplish federated modeling and share global models without sharing data. Federated learning offers a good answer to complex data and privacy security issues. Although there are many ways to target federated learning, Byzantine attacks are the ones we concentrate on. Byzantine attacks primarily impede learning by tampering with the local model parameters provided by a client to the master node throughout the federation learning process, leading to a final global model that diverges from the optimal solution. To address this problem, we combine aggregation rules with Byzantine robustness using a gradient descent optimization algorithm based on variance reduction. We propose a WGM-dSAGA method with Byzantine robustness, called weighted geometric median-based distributed SAGA. We replace the original mean aggregation strategy in the distributed SAGA with a robust aggregation rule based on weighted geometric median. When less than half of the clients experience Byzantine attacks, the experimental results demonstrate that our proposed WGM-dSAGA approach is highly robust to different Byzantine attacks. Our proposed WGM-dSAGA algorithm provides the optimal gap and variance under a Byzantine attack scenario.

Keywords: federated learning; Byzantine attack; gradient descent optimization; parameter aggregation

check for updates

Citation: Wang, X.; Zhang, H.; Bilal, A.; Long, H.; Liu, X. WGM-dSAGA: Federated Learning Strategies with Byzantine Robustness Based on Weighted Geometric Median. *Electronics* **2023**, *12*, 1190. https://doi.org/10.3390/electronics12051190

Academic Editors: Nyothiri Aung, Tao Zhu and Sahraoui Dhelim

Received: 14 February 2023
Revised: 27 February 2023
Accepted: 27 February 2023
Published: 1 March 2023

1. Introduction

The rapid advancement of big data and artificial intelligence has created new opportunities for established sectors to modernize and adapt. Still, it has also raised new concerns about network and data security. Both companies and individuals have more stringent privacy protection needs, and there is an urgent need for technologies that can provide privacy protection to compensate for the deficiencies of distributed learning in joint data and privacy security training. More institutions and people may be encouraged to supply data, which will aid in the growth of linked sectors if privacy-preserving technologies can be developed to ensure that neither party using data for federated training is aware of each other's data. Federated learning was developed under this framework. A decentralized, privacy-preserving approach to data processing is offered by federated learning.

The components of federated learning are a master node server and a number of clients. The clients' data are kept private, and dispersed devices are used to execute local computations. Each client's private data generate local model updates (such as stochastic gradients, corrected stochastic gradients, and model parameters). At the same time, the master node server aggregates the local model variables for global model updates and propagates the aggregated results to the clients [1].

Even though federated learning protects privacy, its dispersed nature makes it prone to errors and adversarial assaults, and devices may become unreliable in terms of computation or communication and may even be hacked by adversaries. A hijacked device may send malicious messages to the master node server when one or more nodes of the individual clients of federated learning experience a problem. This will confuse the learning process

and may cause the global model parameters to fail or converge [2]. Therefore, our attention is directed at the harmful assault known as the Byzantine attack [3].

The formulations of parameter estimates cannot be properly derived in machine learning since some target models' loss functions are extremely complex. Gradient descent is typically employed in practical applications to train models, with the goal having the trend of its training results being decreased to the minimal value. However, the training process of gradient descent is incredibly sluggish and sensitive to hyperparameters, such as learning rate and weight decay. To solve this issue, variance reduction techniques have been found to be an effective application, demonstrating linear convergence speed, whereas other stochastic methods yield sublinear speed [4].

Defensively, geometric median (GM) is extensively employed for gradient aggregation in the research literature on robust distributed learning [5], as it can withstand up to half of the Byzantine clients and estimate the true underlying parameters [6]. In particular, GM exhibits convergence qualities when employed as an aggregation rule in gradient descent algorithms [7]. In more recent investigations, GM has also been used to aggregate parameters to obtain reliable federated learning solutions [8].

Even though current stochastic gradient descent optimization algorithms can guarantee that a model converges to the optimal solution domain in the absence of attacks, this domain will grow when Byzantine attacks are present, and the model's learning will become subpar. On the other hand, existing distributed learning models often use basic aggregation rules, and there is no practical method for dealing with parameters that are uploaded by hostile nodes in the event of Byzantine attacks during the learning process.

The weighted geometric median-based distributed SAGA (WGM-dSAGA) method, which combines variance reduction-based distributed SAGA with robust aggregation rules to improve the Byzantine robustness of federated learning, is suggested in this research based on the issues that currently exist. While conventional distributed SAGA aggregates parameters using mean values, the WGM-dSAGA does this at the master node server using weighted geometric medians. To calculate the distance (Euclidean and cosine) between local parameter changes made by various clients and discover outliers in the distance space, the WGM-dSAGA employs a state-of-the-art parameter-free outlier detection technique called COPOD [9]. After that, it maps the outlier scores to client-specific weights and uses the client-weighted geometric median for global model updates. Experimental evidence is provided to support the robustness of the WGM-dSAGA against various Byzantine attacks. In conclusion, we highlight the following contributions of this paper:

To counter Byzantine attacks, we suggest a novel WGM-dSAGA federated learning technique with Byzantine robustness.

By calculating the Euclidean distance matrix and the cosine distance matrix of client parameters and utilizing the most recent parameter-free outlier detection technique COPOD, the proposed WGM-dSAGA algorithm assesses the maliciousness of client parameters.

The WGM-dSAGA approach also assigns the weight of each client in each iteration based on the maliciousness of the client parameters, reducing the contribution of malicious nodes to global model updates and improving federated learning robustness.

We also thoroughly assess the WGM-dSAGA's performance under three Byzantine attack scenarios, and the results demonstrate that the proposed WGM-dSAGA algorithm performs better than current Byzantine protection methods that are in use.

The remainder of this paper is structured as follows: Section 2 describes the current research progress on gradient descent optimization algorithms and Byzantine-robust aggregation rules. Section 3 details the development of the proposed WGM-dSAGA algorithm. Section 4 provides our experimental results and performance evaluation. Section 5 presents the conclusions and a discussion of future work.

2. Related Works

2.1. Gradient Descent Algorithms

Researchers have consistently improved gradient descent algorithms, which can be thought of as the basic beginning point, compared to the most recent deep learning optimization algorithms, to tackle the optimization problem of loss function in machine learning and deep learning. The goal of applying a gradient descent algorithm is to keep iterating until the loss function value is smaller than the hyperparameter threshold, at which point the algorithm is terminated.

Two primary divisions of gradient descent optimization algorithms are methods based on momentum and variance reduction and algorithms based on adaptive learning rates [10]. Although stochastic gradient descent approaches significantly lower the computational cost by updating the gradient by picking random samples each time, the noise in the gradient estimation at the end of training prevents it from converging further. The stochastic variance reduction gradient method (SVRG) [11] and other algorithms are examples of solutions for this purpose. Without using any reduction techniques, the variance-reduced stochastic gradient descent algorithm (VR-SGD) [12] can handle non-smooth or non-strongly convex problems directly and has linear convergence with a much quicker convergence rate. The new optimization method SAGA [13] has also been proposed. The SAGA enhances the theory underlying the SAG and SVRG with better theoretical convergence speed and enables composite objectives by employing proximal operators on the regulator.

Representative approaches for solving the problem of a non-convex objective function falling into local suboptimality and improving an algorithm's convergence speed include AdaGrad [14], Adadetla [15], RMSprop, Adam [16], and others. Optimization approaches based on adaptive learning rates have higher convergence when training increasingly complicated neural networks. AdaDGS [17] is a straightforward, clever, and effective adaptive technique that optimizes the use of DGS gradients, eliminating the necessity of fine-tuning the hyperparameters. The AdaTerm [18] method derives the first momentum and all the involved statistics based on the t-distribution, thereby possessing adaptive noise capability and improved learning performance.

The variance reduction-based gradient descent optimization technique SAGA is used in this paper through dSAGA, a distributed implementation. For distributed learning application scenarios such as federated learning, where each device analyzes a constrained number of data samples, dSAGA is ideally suitable.

2.2. Byzantine-Robust Aggregation Rules

Based on the goal of the assaults, adversarial attacks in federated learning applications can be broadly divided into two types, namely undirected attacks and directed attacks [19]. The adversary's goal in directed attacks (also known as backdoor attacks) is to cause a model to maintain good overall performance in its primary job while performing poorly in some specified subtasks. An undirected attack aims to confuse a model and prevent it from performing at its best in the main task. Non-directed attacks are easier to spot because they reduce the primary task's overall performance. Since the adversary's objective is frequently unknown in advance, backdoor-directed attacks are more difficult to identify.

Based on the attacker's capabilities, undirected and directed attacks can be further categorized into model attacks and data attacks. A model attack occurs when an attacked client modifies the update parameters of the local model, thereby altering the overall model (e.g., client 1 in Figure 1). A data assault occurs when an attacker can alter a portion of all training samples in a way that the model learner is unaware of (e.g., client 2 in Figure 1).

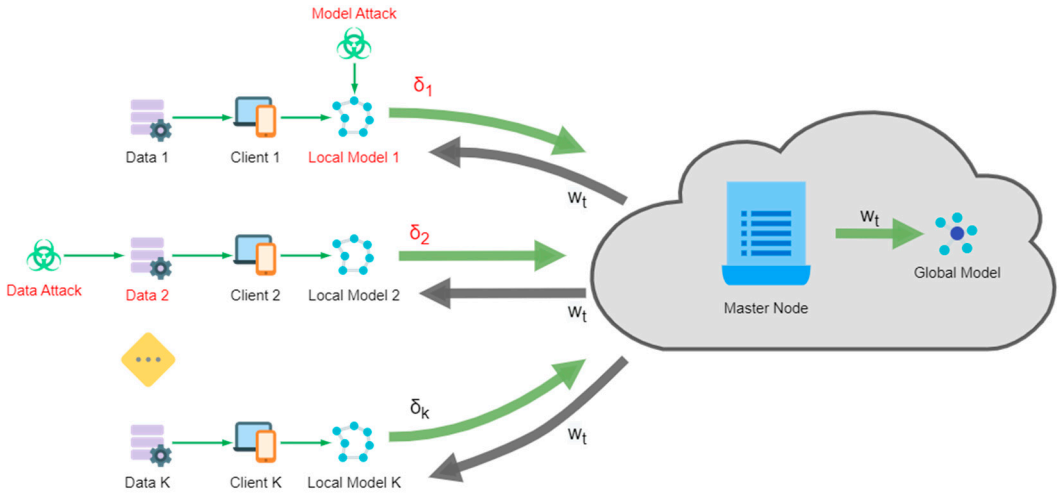

Figure 1. Overview of the federated learning approach, which uses local datasets to train machine learning models across numerous clients. The master node server aggregates the local model parameters provided by the clients and updates the global model. It then broadcasts the updated global model parameters to the clients throughout each training round for coordinated training. Some clients experiencing Byzantine assaults send malicious messages to the master node to obstruct the federation learning process due to the model attacks (such as client 1) and data attacks (such as client 2) on those clients.

Several robust aggregation rules have been developed in recent years, primarily for enhancing distributed stochastic gradient descent (SGD) solvers for the underlying optimization task to deal with Byzantine attacks in federated learning. Stochastic algorithms can withstand a limited number of clients under attack from Byzantine adversaries by aggregating stochastic gradients using aggregation techniques, such as geometric median [5], median [20], trimmed mean [6], or iterative filtering [21]. Other aggregation algorithms include Krum [22], which chooses a random gradient that is closest to the provided number of nearest random gradients with the least cumulative squared distance. RSA [23] combines models besides random gradients by penalizing local and global parameter variations.

Shenghui Li et al. proposed an automatic weighted geometric median (AutoGM) [24] aggregation method that is flexible and robust. To achieve the value of the AutoGM, an algorithm based on an alternating optimization strategy was designed to improve the robustness against model poisoning and data poisoning attacks. To make the method differentially private by utilizing a tree-based mechanism, Ali Jadbabaie et al. suggested a resilient aggregation rule that employs a geometric median resistant to Byzantine assaults with fewer than half of the number of clients [25]. Local malicious factor (LoMar), developed by Naif Xingyu Li et al. [26], is a two-stage defense algorithm that scores model updates for each remote client by determining the relative distribution of its neighbors using a kernel density estimation method and aggregates the local models based on the scores.

When aggregating the local model parameters uploaded by a client on the server side, the weighted geometric median aggregation method described in this research can tolerate Byzantine assaults better than current geometric median-based methods because the weighted parameters are derived from outlier detection and contribute to the solution process of the geometric median, which can significantly lessen the impact of the malicious parameters uploaded by clients that are vulnerable to Byzantine attacks.

3. Proposed Methodology

This section begins by introducing the distributed SAGA with averaging. After that, we suggest a way to obtain client weights using the parameter-free outlier detection algorithm COPOD. Finally, the Weiszfeld algorithm calculates the weighted geometric median in linear time [27]. Our proposed distributed SAGA based on a weighted geometric

median, abbreviated as WGM-dSAGA, substitutes robust aggregation based on a weighted geometric median for the mean aggregation method, as detailed in Algorithm 1.

Algorithm 1: Robust aggregation rules based on weighted geometric medians

Require: learning rate γ; number of clients K; number of data samples N on every client k; number of iterations t;
Master node and honest client initialize $\mathcal{W}^0$
for all honest client k **do**
  for $n \in \{1, 2, \cdots, N\}$ **do**
    Initializes gradient storage $\mathcal{Z}_{k,n}(\phi_{k,n}) = \mathcal{Z}_{k,n}(\mathcal{W}^0)$
  end for
  Initializes average gradient $\overline{\mathcal{G}}_k^1 = \frac{1}{N}\sum_{n=1}^{N}\mathcal{Z}_{k,n}(\mathcal{W}^0)$
  Sends $\overline{\mathcal{G}}_k^1$ to master node
end for
Mater node updates $\mathcal{W}^1 = \mathcal{W}^0 - \gamma \cdot geomed_{k\in\mathcal{K}}\left\{\overline{\mathcal{G}}_k^1\right\}$
for all $t = 1, 2, \cdots$ **do**
  Master node broadcasts $\mathcal{W}^t$ to all clients
  for all honest client node k **do**
    Samples i_k^t from $\{1, \cdots, N\}$ uniformly at random
    Updates $\mathcal{M}_k^t = \mathcal{Z}_{k,i_k^t}(\mathcal{W}^t) - \mathcal{Z}_{k,i_k^t}\left(\phi_{k,i_k^t}\right) + \overline{\mathcal{G}}_k^1$
    Sends $\mathcal{M}_k^t$ to master node
    Updates $\overline{\mathcal{G}}_k^{t+1} = \overline{\mathcal{G}}_k^t + \frac{1}{N}\left(\mathcal{Z}_{k,i_k^t}(\mathcal{W}^t) - \mathcal{Z}_{k,i_k^t}\left(\phi_{k,i_k^t}\right)\right)$
    Stores gradient $\mathcal{Z}_{k,i_k^t}\left(\phi_{k,i_k^t}\right) = \mathcal{Z}_{k,i_k^t}(\mathcal{W}^t)$
  end for
  for $k, l = 1, 2, \cdots, K$ **do**
    Master node updates $\mathcal{D}_{k,l}^{E} = \|\mathcal{M}_k^t - \mathcal{M}_l^t\|_2$, $\mathcal{D}_{k,l}^{C} = 1 - \frac{\mathcal{M}_k^t \mathcal{M}_l^t}{\|\mathcal{M}_k^t\|_2 \|\mathcal{M}_l^t\|_2}$
    Master node updates $\mathbb{D}_E = \left[\mathcal{D}_{k,l}^{E}\right]$, $\mathbb{D}_C = \left[\mathcal{D}_{k,l}^{C}\right]$
    Master node updates $\mathcal{S}_k^E = COPOD(\mathbb{D}_E)$, $\mathcal{S}_k^C = COPOD(\mathbb{D}_C)$
    Master node updates $\mathcal{S}_k^t = \left(\mathcal{S}_k^E + \mathcal{S}_k^C\right)/2$
    Master node updates $\alpha_k^t = \frac{exp\left(-\mathcal{S}_k^t\right)}{\sum_{l=1}^{K} exp\left(-\mathcal{S}_l^t\right)}$
  end for
  Master node updates $\mathcal{W}^{t+1} = \mathcal{W}^t - \gamma \cdot \underset{\mathcal{M}}{\operatorname{argmin}} \sum_{k\in\mathcal{K}} \alpha_k^t \|\mathcal{M} - \mathcal{M}_k^t\|_2$
end for

3.1. Distributed SAGA with Mean Aggregation

We assume a federated learning network with K clients and a single master node (data center). Each client in a distributed SAGA keeps a random gradient table of its local data samples N. The master node transmits to the client k an initialized global model parameter $\mathcal{W}^t$ at iteration t. Each client chooses a local data sample with an index i_k^t at random, after which each client calculates the stochastic gradient $\mathcal{Z}_{k,i_k^t}(\mathcal{W}^t)$ for that particular data sample. In the distributed SAGA, a client must also take into account the historical gradient stored in the local data sample during the gradient update process, rather than sending the random gradient $\mathcal{Z}_{k,i_k^t}(\mathcal{W}^t)$ directly to the master node at this period. The client then saves $\mathcal{Z}_{k,i_k^t}(\mathcal{W}^t)$ as the random gradient of the data sample, which is indexed as i_k^t, and delivers the rectified random gradient to the master node. The master node updates the global model parameters $\mathcal{W}^{t+1}$ once all clients have sent the random gradients to it.

When performing iteration t, client k sets $\phi_{k,n}^{t+1} = \mathcal{W}^t$ and stores the random gradient $\mathcal{Z}_{k,n}\left(\phi_{k,n}^{t+1}\right)$ while client k randomly selects the data samples with index n. It is assumed that the ϕ of the remaining data samples is constant, i.e., $\phi_{k,n}^{t+1} = \phi_{k,n}^{t}$, when $n \neq i_k^t$.

$$\mathcal{G} := \mathcal{Z}_{k,i_k^t}(\mathcal{W}^t) - \mathcal{Z}_{k,i_k^t}\left(\phi_{k,i_k^t}^t\right) + \frac{1}{N}\sum_{n=1}^{N}\mathcal{Z}_{k,n}\left(\phi_{k,n}^t\right) \tag{1}$$

is the adjusted random gradient of client k at iteration t. The following formula is used to update the parameters of the global model:

$$\mathcal{W}^{t+1} = \mathcal{W}^t - \gamma \cdot \frac{1}{K}\sum_{k=1}^{K}\mathcal{G}_k^t \tag{2}$$

where γ stands for the learning rate.

3.2. Client Weights Based on the COPOD Outlier Detection

A brand-new approach for finding outliers, called COPOD, is based on an empirical copula estimation of the tail probability [9]. One of the best outlier identification methods is COPOD, which is deterministic and free of hyperparameters. No random training or learning is used in this algorithm; instead, it is based on empirical cumulative distribution functions (ECDFs). The above features prevent model performance from being harmed by inappropriate hyperparameter selection or other potential biases, while obtaining client-side weights. Additionally, COPOD is an extremely effective outlier identification approach with little computing overhead based on fast execution, making it the perfect choice when working with high-dimensional big datasets.

We suppose that there are B clients in the federation learning network exposed to Byzantine attacks and that these risky clients are unknown to the master node. According to our presumption, $\mathcal{K}$ is the set of all clients, $\mathcal{B}$ is the set of all clients exposed to the Byzantine assault, and the two sets fulfil $|\mathcal{K}| = K$ and $|\mathcal{B}| = B$. We assume that the following criterion $B < \frac{K}{2}$ is always satisfied based on existing mainstream research findings. Instead of sending the correct local model parameters to the master node during federated learning, a client under a Byzantine attack will send malicious messages to the master node. As a result, we modify the algorithm for client k by providing harmful local model parameters to the master node at iterations t:

$$\mathcal{M}_k^t = \begin{cases} \mathcal{G}_k^t, & k \notin B \\ *, & k \in B \end{cases} \tag{3}$$

where $*$ represents the client that is the target of the Byzantine assault.

The Euclidean distance is calculated as follows:

$$\mathcal{D}_{k,l}^{E} = \|\mathcal{M}_k^t - \mathcal{M}_l^t\|_2 \tag{4}$$

The cosine distance is calculated as follows:

$$\mathcal{D}_{k,l}^{C} = 1 - \frac{\mathcal{M}_k^t \mathcal{M}_l^t}{\|\mathcal{M}_k^t\|_2 \|\mathcal{M}_l^t\|_2} \tag{5}$$

Between all local models, the parameters are calculated at iteration t, where $k, l = 1, 2, \cdots, K$, after all clients have uploaded their local model parameters to the master node. The COPOD model is used to compute the $(n \times n)$ shape matrices $\mathbb{D}_E = \left[\mathcal{D}_{k,l}^{E}\right]$ and $\mathbb{D}_C = \left[\mathcal{D}_{k,l}^{C}\right]$, and outlier detection is carried out. We calculate and average the outlier scores $\mathcal{S}_k^E = COPOD(\mathbb{D}_E)$ and $\mathcal{S}_k^C = COPOD(\mathbb{D}_C)$ to arrive at the final outlier score $\mathcal{S}_k^t = \left(\mathcal{S}_k^E + \mathcal{S}_k^C\right)/2$ for client k at iteration t. To obtain each client's weight at iteration T,

we need to normalize the computed outlier scores using the Softmax function. We need to suppress the influence of clients with higher outlier scores on the global model parameter updates and enhance the contribution of clients with lower outlier scores on the global model parameter updates [28]. As a result,

$$\alpha_k^t = \frac{exp(-\mathcal{S}_k^t)}{\sum_{l=1}^{K} exp(-\mathcal{S}_l^t)} \tag{6}$$

is used to determine the clients' weight parameters.

3.3. Robust Aggregation Rules Based on Weighted Geometric Medians

A geometric median is a generalization of the resilient aggregation rule that we suggest. The geometric median $\hat{\mathcal{M}}$ of a vector set $\mathcal{V} = \{\mathcal{M}_k\}_{k\in\mathcal{K}}$ is the point from which the total of the Euclidean distances to all of the points in $\mathcal{V}$ is reduced to the minimum. The formula

$$\hat{\mathcal{M}} := \underset{\mathcal{M}}{\operatorname{argmin}} \sum_{k\in\mathcal{K}} \|\mathcal{M} - \mathcal{M}_k\|_2 \tag{7}$$

is satisfied by the geometric median point $\hat{\mathcal{M}}$. In the Byzantine-robust federated learning research, geometric medians are frequently employed as gradient aggregation rules and exhibit convergence qualities [20]. A geometric median's robustness, however, is constrained since its constant factor is still affected by the presence of outliers [5], meaning that the bound rises as the number of outliers does. We further develop a weighted geometric median (WGM) as a more adaptable, robust aggregation rule based on geometric median. The weighted geometric median of the set of vectors $\mathcal{V}$ is denoted by the letter $\widetilde{\mathcal{M}} = WGM(\mathcal{V})$. The formula

$$\widetilde{\mathcal{M}} := \underset{\mathcal{M}}{\operatorname{argmin}} \sum_{k\in\mathcal{K}} \alpha_k \|\mathcal{M} - \mathcal{M}_k\|_2 \tag{8}$$

can be used to compute the weighted geometric median. We may use the $(1+\epsilon)$-approximate geometric median to provide an approximately linear time solution to classical optimization problems, which often require sublinear time because finding the precise geometric median is extremely costly [29]. We compute the weighted geometric median in the Byzantine-robust aggregation rule based on weighted geometric medians using the Weiszfeld smoothing algorithm.

4. Numerical Experiments

We set up 50 trustworthy clients and $B = 20$ clients experiencing Byzantine attacks during the federated learning process for the experiments. Using three common Byzantine attacks—the Gaussian assault, the sign-flipping attack, and the zero-gradient attack—we evaluated the performance of the proposed WGM-dSAGA scheme [30].

Gaussian attack: A Byzantine client $k \in \mathcal{B}$ that is subjected to a Gaussian attack modifies its $\mathcal{M}_k^t$ using a Gaussian distribution with a mean of $\frac{1}{K-B}\sum_{k'\notin\mathcal{B}} \mathcal{M}_{k'}^t$ and a variance of 30.

Sign-flipping attack: A Byzantine client $k \in \mathcal{B}$, which suffers from a sign-flipping attack, falsifies its uploaded message to $\mathcal{M}_k^t = u \cdot \frac{1}{K-B}\sum_{k'\notin\mathcal{B}} \mathcal{M}_{k'}^t$ and sets $u = -3$ in the experiment.

Zero-gradient attack: A zero-gradient attack on a Byzantine client $k \in \mathcal{B}$ causes it to upload $\mathcal{M}_k^t = -\frac{1}{B}\sum_{k'\notin\mathcal{B}} \mathcal{M}_{k'}^t$ as the parameter information to the master node, and as a result, the master node's total received parameter information is zero.

In our experiments, we used the ϵ-approximate geometric median and set $\epsilon = 1 \times 10^{-5}$. We used two dichotomous datasets, IJCNN1 and COVTYPE, for our experiments. The IJCNN1 dataset has 49,990 training data samples with a dimension $p = 22$, while the COVTYPE dataset has 581,012 training data samples with a dimension $p = 54$.

We compared the SGD, the mini-batch (B)SGD, and the SAGA in the first set of experiments, where the batch size of the BSGD algorithm was 50. The geometric median

and weighted geometric median aggregation rules were combined with these gradient descent optimization algorithms. Compared to the SGD technique, the BSGD has a higher computing overhead but is more effective at reducing the effect of stochastic gradient noise on the overall model. Contrarily, the SAGA is an ideal stochastic gradient descent optimization technique because it keeps the computing overhead comparable to the SGD while reducing the noise of the stochastic gradient.

The performance of the above algorithms on the IJCNN1 and COVTYPE datasets is shown in Figures 2 and 3. In each algorithm, we used a constant step size, modified the step size, and, in the absence of a Byzantine attack, obtained the ideal optimal gap $\mathcal{Z}(\mathcal{W}^t) - \mathcal{Z}(\mathcal{W}^*)$. For all three classical Byzantine attacks, the weighted geometric median exhibits a smaller optimal performance gap $\mathcal{Z}(\mathcal{W}^t) - \mathcal{Z}(\mathcal{W}^*)$. The WGM-dSAGA outperforms the SGD and the BSGD among the three weighted geometric median aggregation methods, whereas the BSGD outperforms the SGD. It is demonstrated through this experimentation that variance reduction strategies significantly impact the Byzantine robustness of federated learning. As can be seen in Figure 2, the distributed SAGA algorithm using the median of ordinary geometry has the smallest optimal gap and variance on the IJCNN1 data set when it is not being attacked. The distributed SAGA using weighted geometric medians all exhibit minimal optimal gaps and variances under the three main Byzantine attack scenarios. On the COVTYPE dataset, it is important to note that the weighted, aggregated median-based BSGD and SAGA methods exhibit comparable optimal gaps, but the SAGA approach converges more quickly.

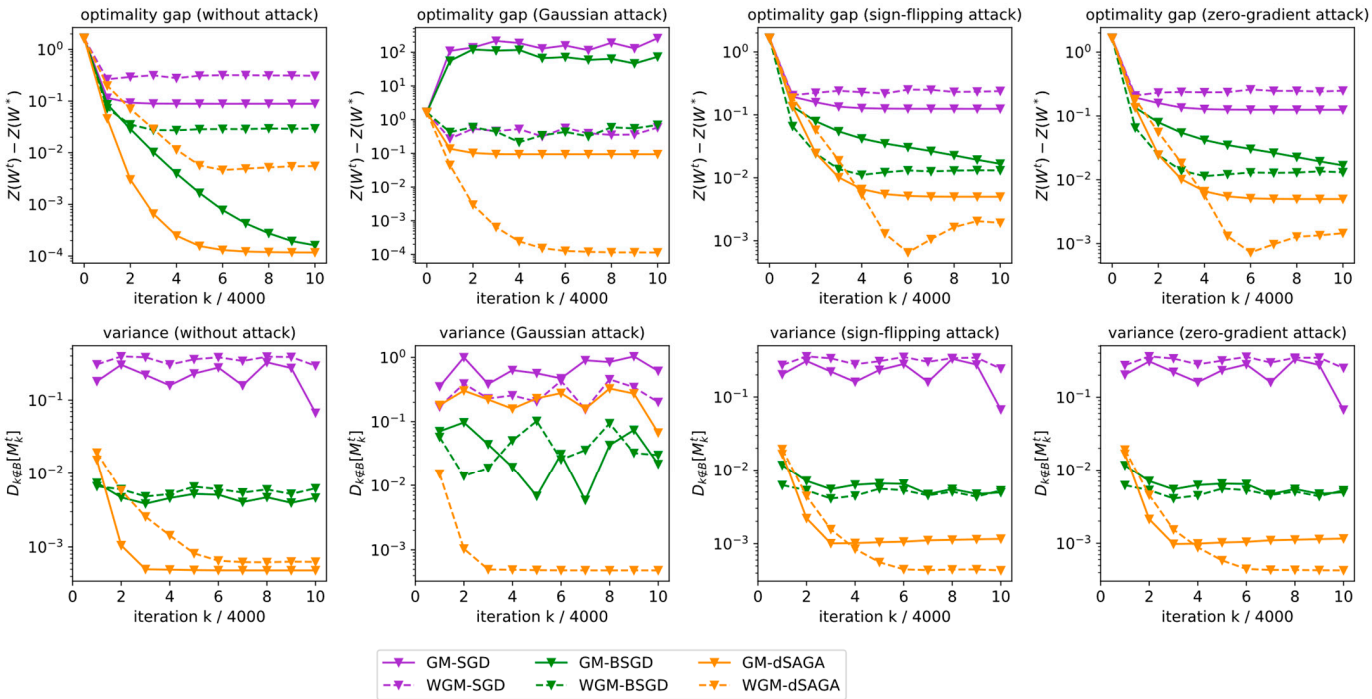

Figure 2. The SGD, mini-batch (B)SGD, and distributed SAGA are combined with geometric median and weighted geometric median and tested for their performance under different attack scenarios on the IJCNN1 dataset, where the step sizes are 0.02, 0.01, and 0.02, respectively. The optimal gap and variation are shown from top to bottom. No attack, a Gaussian attack, a sign-flipping attack, and a zero-gradient attack are represented from left to right.

The SAGA method was then used along with various robust aggregation rules, such as Krum, median, mean, weighted mean, geometric median, and weighted geometric median. The weights using their respective weighted mean and weighted geometric median rules were similarly derived. Figure 4 displays the different techniques' performance on the IJCNN1 and COVTYPE datasets [1]. When there is no Byzantine attack, the distributed

SAGA using weighted mean aggregation has the best optimal gap performance. However, the WGM-dSAGA algorithm, which employs a weighted geometric median, performs admirably in all three Byzantine assault scenarios, including the Gaussian attack, the sign-flipping attack, and the zero-gradient attack, showing the smallest optimal gap.

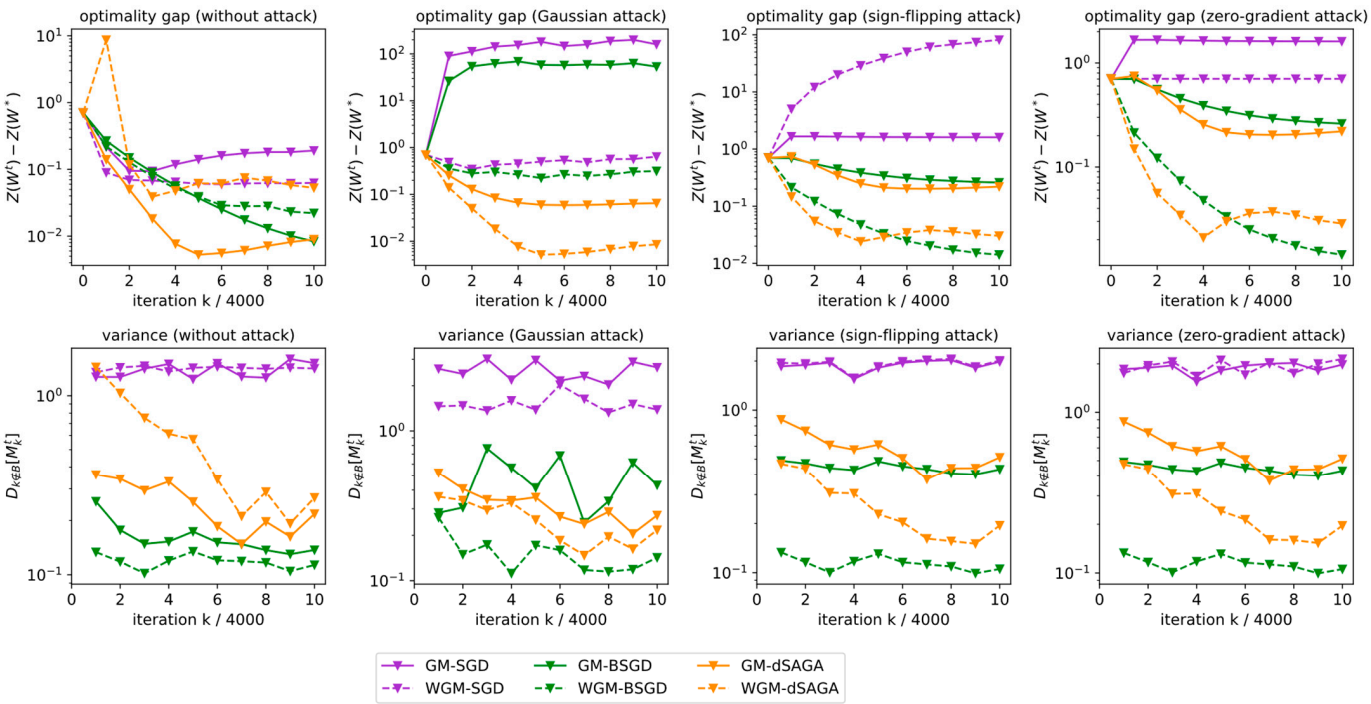

Figure 3. Schemes follow the same formatting. The SGD, mini-batch (B)SGD, and distributed SAGA are combined with geometric median and weighted geometric median and tested for their performance under different attack scenarios on the COVTYPE dataset, where the step sizes are 0.01, 0.0005, and 0.01, respectively. The optimal gap and variation are shown from top to bottom. No attack, a Gaussian attack, a sign-flipping attack, and a zero-gradient attack are represented from left to right.

We conducted multi-classification experiments on the MNIST and CIFAR10 datasets [26]. The MNIST dataset has 60,000 training data samples with a dimension $p = 784$, while the CIFAR10 dataset has 50,000 training data samples with a dimension $p = 3072$. We chose step sizes of 0.1, 0.5, and 0.1 for the SGD, BSGD, and SAGA algorithms, respectively, with a batch size of 50. Tables 1 and 2 display the outcomes of the algorithm runs. The techniques using weighted mean and weighted geometric median aggregation both offer improved accuracy and exhibit better robustness under Byzantine attacks than using weights obtained based on outlier detection. The one with the highest accuracy is the WGM-dSAGA algorithm, which uses a weighted geometric median. It can be seen from Tables 1 and 2, in combination with Figures 5 and 6, that neither gradient optimization descent algorithm nor robust aggregation rules show obvious advantages in the scenario without attacks. Under three different types of Byzantine attack scenarios, the distributed SAGA gradient descent optimization algorithm performs better than the SGD and BSGD algorithms, and the weighted geometric median aggregation rule is better than the other three aggregation rules. It is further proven that the WGM-dSAGA algorithm has better robustness under a Byzantine attack scenario.

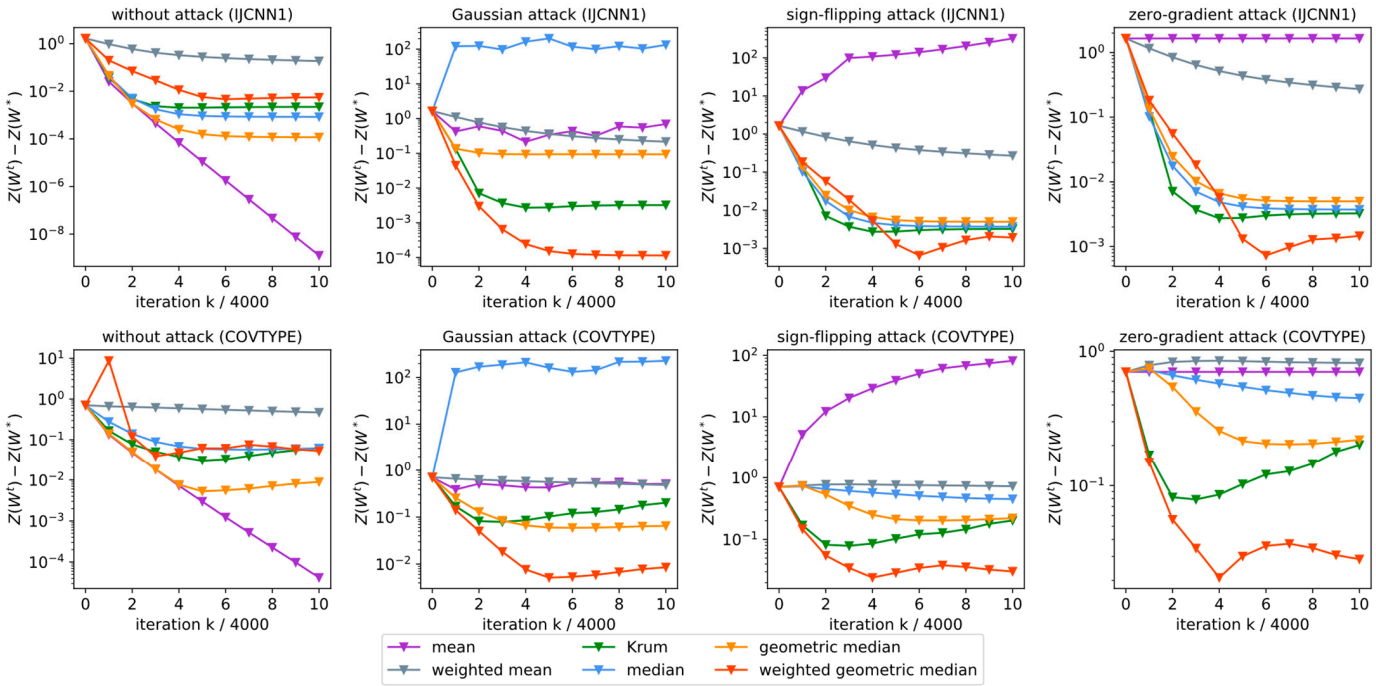

Figure 4. Testing the performance of the distributed SAGA, especially when combined with Krum, median, mean, weighted mean, geometric median, and weighted geometric median, on the IJCNN1 and COVTYPE datasets. The IJCNN1 dataset and the COVTYPE dataset are indicated from top to bottom. No attack, a Gaussian attack, a sign-flipping attack, and a zero-gradient attack are represented from left to right.

Table 1. The SGD, mini-batch (B)SGD, and distributed SAGA are combined with mean, weighted mean, geometric median, and weighted geometric median, respectively, and tested for their accuracy under different attack scenarios on the MNIST dataset. Here, wmean stands for the weighted mean, gm stands for the geometric median, and wgm stands for the weighted geometric median.

Attack	Algorithm	Mean Acc (%)	Wmean Acc (%)	Gm Acc (%)	Wgm Acc (%)
without	SGD	86.1	89.2	82.8	88.2
	BSGD	85.6	83.9	86.0	89.3
	SAGA	87.2	89.7	88.3	86.9
Gaussian	SGD	16.2	18.9	91.9	93.6
	BSGD	27.3	29.6	92.0	93.7
	SAGA	14.5	19.4	91.4	95.9
sign-flipping	SGD	0.12	10.3	0.02	10.6
	BSGD	0.16	11.6	82.3	94.3
	SAGA	0.12	14.8	86.4	95.7
zero-gradient	SGD	9.07	21.6	26.3	32.4
	BSGD	9.83	24.5	81.5	94.1
	SAGA	9.82	25.2	87.4	91.7

Table 2. The SGD, mini-batch (B)SGD, and distributed SAGA are combined with mean, weighted mean, geometric median, and weighted geometric median, respectively, and tested for their accuracy under different attack scenarios on the CIFAR10 dataset. Here, wmean stands for the weighted mean, gm stands for the geometric median, and wgm stands for the weighted geometric median.

Attack	Algorithm	Mean Acc (%)	Wmean Acc (%)	Gm Acc (%)	Wgm Acc (%)
without	SGD	56.2	58.2	51.4	55.1
	BSGD	55.9	57.8	54.5	53.2
	SAGA	54.3	57.7	57.1	56.7
Gaussian	SGD	31.4	34.3	37.1	47.8
	BSGD	30.2	33.7	42.5	54.7
	SAGA	28.4	34.9	43.7	55.8
sign-flipping	SGD	0.35	4.79	1.75	21.3
	BSGD	0.23	9.23	26.4	45.7
	SAGA	0.12	9.71	29.5	55.4
zero-gradient	SGD	1.22	8.91	14.6	17.5
	BSGD	2.37	8.94	32.3	48.5
	SAGA	2.53	9.43	37.1	57.2

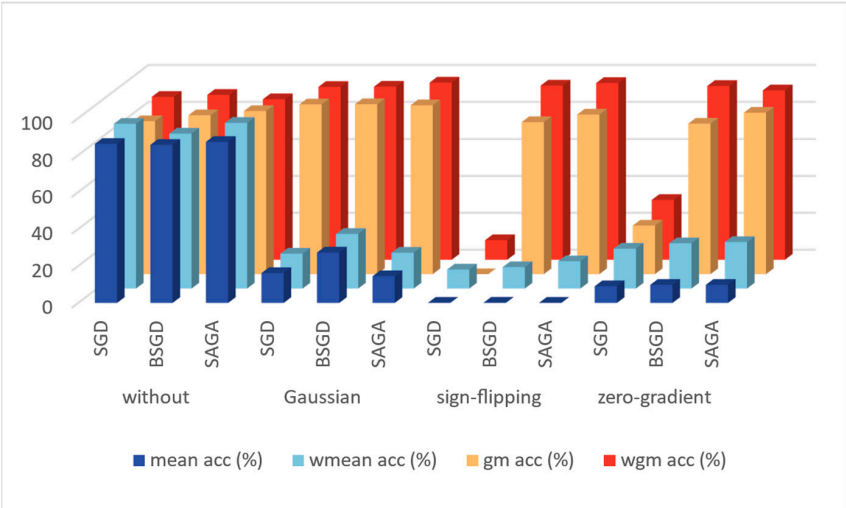

Figure 5. Three-dimensional histogram generated from Table 1.

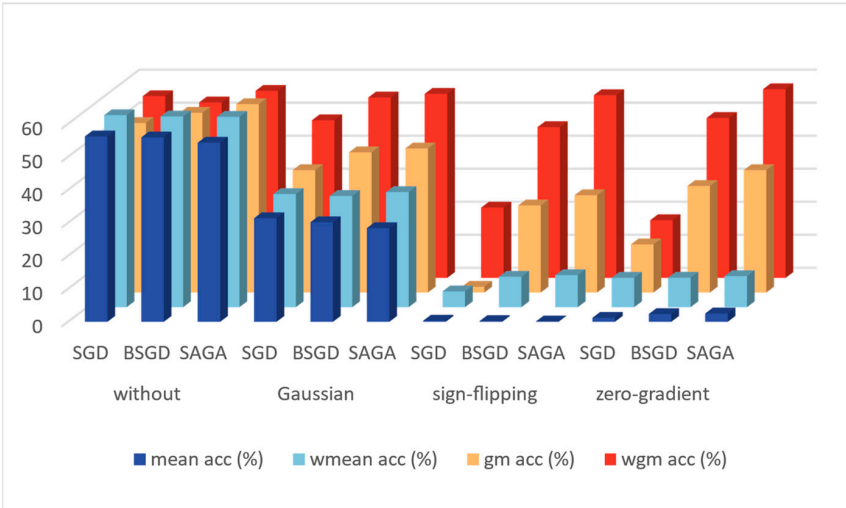

Figure 6. Three-dimensional histogram generated from Table 2.

5. Conclusions

In this paper, we propose a federated learning strategy with Byzantine robustness called the WGM-SAGA. We performed an experiment with the WGM-dSAGA algorithm under three classical Byzantine attack scenarios. Based on the Byzantine problem research background, we assumed that no more than half of the clients would suffer from Byzantine attacks in the experiment. The experimental results show that in the absence of Byzantine attacks, the WGM-dSAGA algorithm performs similarly to existing methods. However, when subjected to Byzantine attacks, the WGM-dSAGA algorithm exhibits excellent Byzantine robustness, provides the optimal gap and variance among all the methods, and significantly improves the convergence speed of the federated learning model. In the multi-classification experiment, we used the same neural network for training and compared the mean and geometric median clustering techniques before and after weighting. The results show that adding our suggested weight calculation method to the local model's parameter aggregation process can effectively increase the accuracy of the model under Byzantine attack scenarios, and the WGM-dSAGA algorithm described in this work has the highest accuracy under Byzantine attacks, which validates its excellent theory of discrimination. The feature extraction of gradients uploaded by clients experiencing Byzantine attacks will be the main topic of our upcoming study in federation learning. We aim to further improve the WGM-dSAGA algorithm's malicious node identification and lessen the impact of bad messages on the entire model.

Author Contributions: Conceptualization and methodology, X.W., H.L. and X.L.; Software, X.W. and H.Z.; Validation, X.W. and H.Z.; Formal analysis, X.L. and A.B.; Writing—original draft X.W.; Writing—review and editing, A.B.; Project administration, H.L.; Funding acquisition, H.L. and X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No. 62262019, No. 61762034, No. 62162025, No. 61966013), the Hainan Provincial Natural Science Foundation of China (No. 620MS046, No. 721QN0890, No. 621QN241, No. 620RC602), the Hainan Provincial key research and development plan of China (No. ZDYF2021GXJS200), the Hainan Provincial reform in education project of China (No. Hnjg2020-31, No. Hnjg2021ZD-15), and the Hainan Provincial Innovative research project for postgraduates of China (No.Qhys2021-305).

Data Availability Statement: The datasets can be downloaded at "https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/ (accessed on 8 October 2022)".

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

References

1. Wu, Z.; Ling, Q.; Chen, T.; Giannakis, G.B. Federated Variance-Reduced Stochastic Gradient Descent with Robustness to Byzantine Attacks. *IEEE Trans. Signal Process.* **2020**, *68*, 4583–4596. [CrossRef]
2. Yang, Z.; Gang, A.; Bajwa, W.U. Adversary-Resilient Distributed and Decentralized Statistical Inference and Machine Learning: An Overview of Recent Advances Under the Byzantine Threat Model. *IEEE Signal Process. Mag.* **2020**, *37*, 146–159. [CrossRef]
3. Dolev, D.; Lamport, L.; Pease, M.C., III; Shostak, R.E. *The Byzantine Generals*; Van Nostrand Reinhold Co.: New York, NY, USA, 1987.
4. Calauzènes, C.; Le Roux, N. Distributed SAGA: Maintaining linear convergence rate with limited communication. *arXiv* **2017**, arXiv:1705.10405.
5. Chen, Y.; Su, L.; Xu, J. Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent. *ACM SIGMETRICS Perform. Eval. Rev.* **2018**, *46*, 96. [CrossRef]
6. Yin, D.; Chen, Y.; Ramchandran, K.; Bartlett, P. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018.
7. Pillutla, K.; Kakade, S.M.; Harchaoui, Z. Robust Aggregation for Federated Learning. *IEEE Trans. Signal Process.* **2022**, *70*, 1142–1154. [CrossRef]
8. Fang, M.; Cao, X.; Jia, J.; Gong, N.Z. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In Proceedings of the USENIX Security Symposium, Boston, MA, USA, 12–14 August 2020.
9. Li, Z.; Zhao, Y.; Botta, N.; Ionescu, C.; Hu, X. COPOD: Copula-Based Outlier Detection. In Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM), Sorrento, Italy, 17–20 November 2020; pp. 1118–1123.
10. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**, arXiv:1609.04747.

11. Yu, T.; Liu, X.-W.; Dai, Y.-H.; Sun, J. Stochastic Variance Reduced Gradient Methods Using a Trust-Region-Like Scheme. *J. Sci. Comput.* **2021**, *87*, 5. [CrossRef]
12. Shang, F.; Zhou, K.; Liu, H.; Cheng, J.; Tsang, I.W.; Zhang, L.; Tao, D.; Jiao, L. VR-SGD: A Simple Stochastic Variance Reduction Method for Machine Learning. *IEEE Trans. Knowl. Data Eng.* **2020**, *32*, 188–202. [CrossRef]
13. Defazio, A.; Bach, F.; Lacoste-Julien, S. SAGA: A Fast Incremental Gradient Method with Support for Non-Strongly Convex Composite Objectives. *Adv. Neural Inf. Process. Syst.* **2014**, *2*, 1646–1654.
14. Duchi, J.; Hazan, E.; Singer, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
15. Zeiler, M.D. ADADELTA: An Adaptive Learning Rate Method. *Comput. Sci.* **2012**. [CrossRef]
16. Kingma, D.; Ba, J. Adam: A Method for Stochastic Optimization. *Int. Conf. Learn. Represent.* **2014**. [CrossRef]
17. Tran, H.; Zhang, G. AdaDGS: An adaptive black-box optimization method with a nonlocal directional Gaussian smoothing gradient. *arXiv* **2020**, arXiv:2011.02009.
18. Ilboudo, W.E.L.; Kobayashi, T.; Matsubara, T. AdaTerm: Adaptive T-Distribution Estimated Robust Moments towards Noise-Robust Stochastic Gradient Optimizer. *arXiv* **2022**, arXiv:2201.06714. [CrossRef]
19. Zhou, X.; Xu, M.; Wu, Y.; Zheng, N. Deep Model Poisoning Attack on Federated Learning. *Future Internet* **2021**, *13*, 73. [CrossRef]
20. Xie, C.; Koyejo, O.; Gupta, I. Generalized Byzantine-tolerant SGD. *arXiv* **2018**, arXiv:1802.10116.
21. Su, L.; Xu, J. Securing Distributed Machine Learning in High Dimensions. *arXiv* **2018**, arXiv:1804.10140.
22. Blanchard, P.; Mhamdi, E.; Guerraoui, R.; Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. In Proceedings of the Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
23. Li, L.; Xu, W.; Chen, T.; Giannakis, G.B.; Ling, Q. RSA: Byzantine-Robust Stochastic Aggregation Methods for Distributed Learning from Heterogeneous Datasets. *arXiv* **2018**, arXiv:1811.03761. [CrossRef]
24. Li, S.; Ngai, E.; Voigt, T. Byzantine-Robust Aggregation in Federated Learning Empowered Industrial IoT. *IEEE Trans. Ind. Inform.* **2023**, *19*, 1165–1175. [CrossRef]
25. Jadbabaie, A.; Li, H.; Qian, J.; Tian, Y. Byzantine-Robust Federated Linear Bandits. *arXiv* **2022**, arXiv:2204.01155.
26. Li, X.; Qu, Z.; Zhao, S.; Tang, B.; Lu, Z.; Liu, Y. LoMar: A Local Defense Against Poisoning Attack on Federated Learning. *arXiv* **2022**, arXiv:2201.02873. [CrossRef]
27. Weiszfeld, E.; Plastria, F. On the point for which the sum of the distances to n given points is minimum. *Ann. Oper. Res.* **2008**, *167*, 7–41. [CrossRef]
28. Alkhunaizi, N.; Kamzolov, D.; Takáč, M.; Nandakumar, K. Suppressing Poisoning Attacks on Federated Learning for Medical Imaging. In Proceedings of the Medical Image Computing and Computer Assisted Intervention—MICCAI 2022, Singapore, 18–22 September 2022; pp. 673–683.
29. Cohen, M.B.; Tat Lee, Y.; Miller, G.; Pachocki, J.; Sidford, A. Geometric Median in Nearly Linear Time. *arXiv* **2016**, arXiv:1606.05225.
30. Lin, F.; Ling, Q.; Xiong, Z. Byzantine-resilient Distributed Large-scale Matrix Completion. In Proceedings of the ICASSP 2019—2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 8167–8171.