applied sciences

MDPI

Article

Efficient and Secure Federated Learning for Financial Applications

Tao Liu [1], Zhi Wang [2], Hui He [3,*], Wei Shi [3], Liangliang Lin [3], Ran An [3] and Chenhao Li [3,4]

[1] Business School, China University of Political Science and Law, Beijing 100088, China
[2] The School of Software, Xi'an Jiaotong University, Xi'an 710049, China
[3] The School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China
[4] Ant Rongxin (Chengdu) Network Technology Co., Ltd., Chengdu 610000, China
* Correspondence: huihe@xjtu.edu.cn

Abstract: Conventional machine learning (ML) and deep learning approaches require sharing customers' sensitive information with an external credit bureau to generate a prediction model, thereby increasing the risk of privacy leakage. This poses a significant challenge for financial companies. To address this challenge, federated learning has emerged as a promising approach to protect data privacy. However, the high communication costs associated with federated systems, particularly for large neural networks, can be a bottleneck. To mitigate this issue, it is necessary to limit the number and size of communications for practical training of large neural structures. Gradient sparsification is a technique that has gained increasing attention as a method to reduce communication costs, as it updates only significant gradients and accumulates insignificant gradients locally. However, the secure aggregation framework cannot directly employ gradient sparsification. To overcome this limitation, this article proposes two sparsification methods for reducing the communication costs of federated learning. The first method is a time-varying hierarchical sparsification method for model parameter updates, which addresses the challenge of maintaining model accuracy after a high sparsity ratio. This method can significantly reduce the cost of a single communication. The second method is to apply sparsification to the secure aggregation framework. Specifically, the encryption mask matrix is sparsified to reduce communication costs while protecting privacy. Experiments demonstrate that our method can reduce the upload communication costs to approximately 2.9% to 18.9% of the conventional federated learning algorithm under different non-IID experiment settings when the sparsity rate is 0.01.

Keywords: federated learning; gradient sparsification; secure aggregation; optimize communication costs

Citation: Liu, T.; Wang, Z.; He, H.; Shi, W.; Lin, L.; An, R.; Li, C. Efficient and Secure Federated Learning for Financial Applications. *Appl. Sci.* **2023**, *13*, 5877. https://doi.org/10.3390/app13105877

Academic Editor: Byung-Gyu Kim

Received: 3 March 2023
Revised: 25 April 2023
Accepted: 28 April 2023
Published: 10 May 2023

1. Introduction

Financial institutions often need to analyze large datasets stored across multiple servers or devices. A common solution is to merge the datasets into a central database [1], which can present several privacy challenges. For instance, the institution may not have the necessary authority or permission to transfer locally stored information, the data owner may not want it shared, and the centralization of the data may increase the severity of a data breach. To address these concerns, Federated Learning (FL) [2,3] has emerged as a promising approach that enables users to jointly generate a global model without explicitly sharing local private data. However, FL can be hindered by high communication costs, as a significant number of devices send local updates to the central server [4]. The aggregation model iteration depends on frequent communication between numerous clients and a single server. Since there is a bandwidth resource limitation, it becomes essential to utilize computing and communication resources efficiently to achieve optimal learning performance in FL [5,6].

Compared to distributed learning, FL is susceptible to communication disruptions between computing nodes. Consequently, the size of the parameter update vector transmitted during aggregation becomes a crucial factor that influences the efficiency of FL. A

larger vector size leads to a longer communication time and increases the average time interval between two parameter aggregation operations on the server [6]. Furthermore, longer transmission times increase the likelihood of facing abnormal situations such as disconnection and network fluctuations, which raises the robustness requirements of the FL algorithm. Therefore, to ensure high communication efficiency, updates should be sent in a compressed and infrequent manner. Gradient sparsification [7] is a commonly used method for gradient compression to reduce communication costs. The main idea of gradient sparsification is to reduce the number of non-zero values in the gradient updates by setting small or insignificant values to zero while retaining the important information for the model updates. This approach can reduce the size of updates and the frequency of communication between clients and the server, resulting in faster convergence and lower communication overhead in FL. During each communication, the data volume transmitted is the same regardless of whether it is transmitting the change value of a small or large parameter. However, parameter updates with larger changes have a greater impact on the performance of the updated model. Therefore, only parameters with large changes are transmitted during parameter update transmission while parameters with small changes are temporarily stored in local accumulation [8]. These parameters are then transmitted to the server once the accumulated value reaches a certain threshold. The advantage of this approach is that each communication transmits important parameter updates with reduced communication volume.

However, the current sparse method still has some limitations. A deep neural network derives its powerful learning ability from its deep structure, where each layer performs a specific function. For instance, in convolutional neural networks (CNN) used for image recognition, each CNN layer corresponds to different feature extraction capabilities. As the number of network layers increases, the extracted features become more and more abstract. Moreover, the parameters of each deep neural network layer have distinct characteristics, with the numerical values differing by orders of magnitude. Therefore, when the model parameters are flattened into a one-dimensional vector, the sparsification process could result in smaller parameter values being covered by larger ones. However, the local update of the local model, including both the feature extraction and classifier layers, in a round may seem insignificant, but they still play an important role during the training process. The reason is that the residual of the local update not uploaded can accumulate over time, causing a delay. Such a delay can eventually deviate the iteration direction from the required descent direction of the current model parameters [9]. If this delay accumulates over multiple rounds, it can cause significant issues. When larger parameters always cover smaller parameters, sparsification can lead to a greater loss. Therefore, it is crucial to consider the impact of sparsification and ensure that smaller parameters are not neglected in the updating process.

Another primary concern for FL systems is protecting user privacy. A series of studies [10,11] have demonstrated that FL can leak sensitive information through the sharing of intermediate model updates. To address this issue, FL commonly integrates Differential Privacy (DP) [12], Homomorphic Encryption (HE) [13], and Secure Multiparty Computation (SMC) [14] for model training, which ensures that the transmitted model information remains confidential. HE enables aggregations to be performed on encrypted data [15,16]; however, it is computationally intensive to execute computations in the encrypted domain. SMC can be used to securely aggregate local model updates without leaking them but it also suffers from the communication bottleneck typical of conventional distributed training. Bonawitz et al. (2017) proposed Secure Aggregation (SA) [17], which permits a server to calculate the sum of large data vectors held by mobile devices in a secure manner without disclosing each user's individual contribution. Nevertheless, implementing SA necessitates significant additional resources in terms of communication and computing to ensure privacy protection.

The challenge of Federated Learning (FL) lies in building a protocol that ensures privacy protection and computational and communication efficiency, without significantly

affecting accuracy. In this paper, we propose an efficient and secure FL framework based on gradient and mask sparsification, which comprises two components: time-varying hierarchical gradient sparsification and encryption mask sparsification. The time-varying hierarchical gradient sparsification algorithm balances the numerical size differences of the network parameters at different layers, reducing the loss caused by the sparsification process. Additionally, we sparsify the encryption mask in the secure aggregation framework, reducing the total amount of data sent, thereby improving communication efficiency and ensuring security. Our contributions can be summarized as follows:

1. We propose an efficient and secure FL framework based on gradient and mask sparsification, which addresses the challenges of privacy protection and computational and communication efficiency in FL.
2. We introduce a time-varying hierarchical gradient sparsification algorithm that balances the numerical size differences of the network parameters at different layers, thereby reducing the loss caused by sparsification. Moreover, we propose to sparsify the encryption mask in the secure aggregation framework to reduce the total amount of data sent, improve communication efficiency, and ensure security.
3. Our experiments on the MNIST, Fashion-MNIST, and CIFAR-10 datasets prove that our method can reduce the upload communication costs to approximately 2.9% to 18.9% of the conventional FL algorithm when the sparsity rate is 0.01.

The structure of this work is organized as follows. Section 2 provides an introduction to the related works and current scholarly advancements in gradient sparsification and secure aggregation. In Section 3, we present an efficient and secure federated learning framework that incorporates time-varying hierarchical gradient sparsification and encryption mask sparsification. Section 4 outlines the experimental settings and methodology, including the results. Finally, in Section 5, we present our conclusions regarding the proposed framework.

2. Related Works

2.1. Gradient Sparsification

In the field of FL communication optimization, numerous approaches have been suggested to address the communication bottleneck issue. Zinkevich et al. (2010) [18] proposed an asynchronous update system that allows each node to independently update the model, free from the influence of other nodes with slower update rates, from a system perspective. In the area of communication transmission, reducing the traffic for each round is a direct method used to decrease communication time. Gradient sparsification, a prevalent gradient compression method, is employed to decrease the communication burden [19].

Strom et al. (2015) [20] proposed a gradient-dropping method, which only sends gradients larger than a predefined constant threshold, and all other gradients are accumulated in the locally saved residuals and are temporarily not sent. This method can achieve up to 3 orders of magnitude compression ratio of uploaded data. However, in the actual training task, it is difficult to choose the appropriate values for the threshold. Due to the different thresholds for the different tasks and models, artificial scheduling can introduce significant experimental costs. In order to overcome this problem, Dryden et al. (2016) [21] enhanced Strom's sparsification method. In contrast to the fixed threshold of the Storm method, the sparsity rate s is fixed. For example, when $s = 0.001$, only the top 0.1% of the gradient update vector is transmitted and other parts of the gradient update are kept locally for the time being. Their experiments showed that with a sparsity rate of $s = 0.001$, their method could only slightly reduce the convergence speed and the final accuracy of the speed training model.

Lei Ba et al. (2016) [22] argued that normalizing each layer is crucial for gradient convergence. Lin et al. (2018) [23] proposed the Deep Gradient Compression (DGC) algorithm, which includes momentum correction, a local gradient client, momentum factor masking, and warm-up training, to address the issue of gradient redundancy in distributed training.

This algorithm can achieve a compression ratio of 270× to 600× without compromising accuracy. Felix Sattler et al. (2019) [8] proposed the Sparse Ternary Compression (STC) framework to compress the locality of deeply compressed Top-k gradient compression and optimal Golomb coding, effectively reducing communication costs in non-IID FL scenarios. Han et al. [24] proposed a fairness-aware gradient descent method that ensures all clients provide an equal number of updates. Additionally, they proposed a novel online learning formulation and algorithm that can determine the near-optimal communication and computational trade-off while minimizing the overall training time. This trade-off is controlled by the degree of gradient sparsity. Qiu et al. [25] presented the first study on the unique issues that arise when introducing sparsity at training time in FL workloads. They then proposed ZeroFL, a framework that relies on highly sparse operations to accelerate on-device training. Some studies quantified the sparse gradient based on gradient sparsification [23,26], ensuring that non-zero element values in the updated gradient vector belonged to a preset numerical set. Subsequently, the parameter update vector could be coded based on the parameter update vector's sparsity to further reduce transmission volume. However, although the above-mentioned works significantly reduced the communication overhead, there was an obvious impact on the model performance. Moreover, privacy issues arose as they directly exposed the model updates to other entities.

2.2. Secure Aggregation

The secure aggregation framework utilizes the Diffie–Hellman (DH) protocol for key exchange [27]. Multiple participants exchange public keys and upon completion of local training, the gradient is not directly transmitted. Instead, a mask matrix, equal in size to the gradient matrix, is generated according to the DH protocol's public key. This random mask is then added to the original gradient to mask the actual gradient information [28,29]. As a result, the server cannot obtain the participants' gradient information. However, the secure aggregation framework faces communication bottlenecks since the gradient sparsification method cannot be directly applied to protect user privacy.

The inherent shortcomings of these methods motivated us to find a better way to combine gradient sparsification with the secure aggregation algorithm. In general, there are two ways to achieve this goal, including sparsing the encrypted gradient and encrypting the spared gradient.

2.2.1. Sparse the Encrypted Gradient

After local training, each participant calculates the gradient information to be sent and uses the gradient sparsification method for Top-k selection to record the location of the gradient to be sent. Subsequently, the encrypted mask is added to the original gradient, and then only the information at the position corresponding to the record on the masked gradient is sent according to the previous record. In this way, the gradient is protected and the server cannot collect the real gradient information. However, there is a problem with this method. Even if each participant has a relatively similar trend toward the descending direction of the gradient, it cannot guarantee that all Top-k positions will completely overlap in each round of training. As a result, when a participant encrypts a specific part of the gradient vector, another participant holding the symmetric mask may not have the corresponding Top-k gradient at that location. Therefore, the encryption mask at that location will not be sent and the mask cannot be eliminated in the aggregation process. The mask that cannot be eliminated is equivalent to adding noise to a component position of the gradient vector. If the influence of the mask is too large, it will have a great impact on the overall convergence direction, as well as the convergence speed and accuracy, of the global model. In the case of multi-party participation, this situation may be more serious and it is not feasible to directly sparsify the encrypted gradient vector.

2.2.2. Encrypt the Spared Gradient

If all positions of the local sparse gradient vector are mask encrypted and sent, this method can ensure that the mask can be eliminated. However, this global sparse gradient encryption scheme violates the original intention of gradient sparsification because the encrypted mask will cover the communication costs reduced by gradient sparsification. If we want to achieve this level of encryption, the communication costs will be huge. If the encrypted gradients are float numbers, the number of encrypted gradients sent by each participant is no different from the traditional security aggregation framework. At the same time, due to the gradient sparsification method in the local calculation process, this scheme will undoubtedly increase the training time compared to the original scheme, which is a negative optimization scheme.

Compared to the previous works, our proposed method can achieve efficient computation and communication for secure aggregation, without compromising accuracy.

3. Efficient and Secure Federated Learning Based on Gradient and Mask Sparsification

In this section, we focus on the detailed designs of the proposed efficient and secure FL framework. Firstly, we propose a time-varying hierarchical sparse method for updating the model parameters. This method allows participants to send only critical updates when communicating with the server, thereby significantly reducing communication costs. Secondly, we combine the characteristics of gradient sparsification and the secure aggregation framework to ensure that the effect of gradient sparsification on communication optimization is not masked by the encryption mask matrix. Our joint design aims to optimize communication efficiency while maintaining data security.

3.1. Time-Varying Hierarchical Gradient Sparsification

The hierarchical gradient sparsification of the network can reduce the loss caused by the sparsification process. In the early stages of training, network parameters usually undergo significant changes. However, after a certain number of update iterations, the amplitude of the changes decreases. Capitalizing on this feature, we propose a time-varying hierarchical gradient sparsification (THGS) algorithm. This algorithm builds upon hierarchical sparsification and reduces the sparsity rate with an increase in iteration rounds. Eventually, it reaches a set lower bound, as illustrated in Algorithm 1.

Each participant in THGS calculates a model update, denoted as g, based on its local private data. Assuming that the model update of the i-th layer is g_i, THGS first identifies the Top-k largest values of g_i. At the same time, THGS sets a threshold δ to the k-th largest value in g_i. Then, THGS calculates a mask vector $\widetilde{g}_i$ for sparsification. Specifically, we set the element to 1 in $\widetilde{g}_i$ if the value on the corresponding index of g_i is greater than the threshold δ. Conversely, if the corresponding value of g_i is smaller than the threshold δ, the element is set to 0 in $\widetilde{g}_i$. Next, THGS computes a sparse model update $g_{i-sparse}$ using the Hadamard product of g_i and $\widetilde{g}_i$, namely, $g_{i-sparse} = g_i \odot \widetilde{g}_i$. Simultaneously, THGS needs to calculate a residual model update $g_{i-residual}$ of g_i, which will be kept locally and accumulated in the subsequent communication rounds. Specifically, $g_{i-residual} = g_i - g_{i-sparse}$. Eventually, each participant uploads the sparse model update $g_{i-sparse}$ to the server for further aggregation, which significantly reduces the upload communication overhead.

We assume that the initial sparsity rate is s_0, α is a constant attenuation factor, and the lower limit of the sparsity rate is s_{min}. In a deep neural network, the sparsity rate s_i of the i-th layer parameter is:

$$s_i = \begin{cases} s_0, & i = 1 \\ s_{i-1} \cdot \alpha, & if \quad s_{i-1} \cdot \alpha > s_{min} \\ s_{min}, & else \end{cases} \tag{1}$$

Algorithm 1: THGS algorithm.

Input: $w, L, s_0, s_{min}, \alpha, g_i$
Output: $w_{sparse}, w_{residual}$
$w_{sparse} \leftarrow torch.zeros\_like(w)$;
$w_{residual} \leftarrow torch.zeros\_like(w)$;
for $(i = 1, i \leq L, i++)$ **do**
  $\widetilde{g_i} \leftarrow to\_vec(abs(g_i))$;
  $k \leftarrow int(len(\widetilde{g_i}) \times s_i)$;
  $\delta \leftarrow TopK(\widetilde{g_i}, k)$;
  $zeros \leftarrow torch.zeros\_like(g_i)$;
  $ones \leftarrow torch.ones\_like(g_i)$;
  $\widetilde{g_i} \leftarrow torch.where(\widetilde{g_i} \leq \delta, zeros, \widetilde{g_i})$;
  $\widetilde{g_i} \leftarrow torch.where(\widetilde{g_i} > \delta, ones, \widetilde{g_i})$;
  $g_{i-sparse} \leftarrow \widetilde{g_i} \odot g_i$;
  $g_{i-residual} \leftarrow g_i - g_{i-sparse}$;
  $w_{sparse} \leftarrow g_{i-sparse}$;
  $w_{residual} \leftarrow g_{i-residual}$;

The process of sparsification can enhance the security of gradient updates in Federated Learning (FL). In FL, gradient attacks can be conducted by the server, which takes the simulated sample data as the input and the gradient sent by the client as the label to infer the true input data. This can be achieved by taking the difference between the updated gradient after the simulated sample data are sent to the initial model for training and the updated gradient actually sent by the client as the penalty function. The simulated sample data are then continuously optimized until the gradient calculated by the simulated sample data and the gradient sent by the client reach a close degree of similarity. By doing this, the simulated sample data will be similar to the local sample data of the client and the original value of the client's sample will be obtained.

The server can only upload a small percentage of the real gradient data such as one percent or one thousandth, which weakens the server's ability to carry out a gradient attack. The label itself, as a penalty function, is incomplete data, making the simulated sample fitted by this method far from the real sample. Sparsification can further reduce the amount of data uploaded to the server, which ultimately weakens the server's ability to carry out a successful gradient attack.

3.2. Sparsify the Encryption Mask for Secure Aggregation

The purpose of applying the gradient sparsification method to the secure aggregation framework is to reduce the cost of communication by sending gradients in important directions while protecting user data privacy. Whether the secure aggregation framework with gradient sparsification is effective depends on the following two conditions:

1. After aggregating the encrypted gradient updates of all participants, the server must ensure that the masks used by participants for local data security are removed from the final aggregation results after combining the encrypted gradient updates;
2. The amount of data transmitted during communication is significantly reduced when using the secure aggregation framework with gradient sparsification compared to the framework without gradient sparsification.

We propose a method to calculate the encryption mask matrix with zero local value, as shown in Algorithm 2, to prevent the masking effect from concealing the impact of gradient sparsification on communication optimization.

Each participant k calculates a local model update at round t, denoted as G_t^k, based on its local private data and the previous residual local update. We first generate a shared mask matrix $mask_r$ between every two participants. Assuming that the local update of the

i-th layer is $G_t^k[i]$, the Top-k largest values of $G_t^k[i]$ are identified, where $k = \max(R_k, R_{min})$. In addition, we set a threshold δ to the k-th largest value in $G_t^k[i]$. Then, we generate a random uniform distribution mask matrix $mask_e$ filtered by σ, where σ is a random encryption mask filtering threshold. Specifically, if a mask value on a specific index of $mask_r$ is greater than the threshold σ, we set the value to 0. Conversely, the value is retained. The purpose is to minimize the impact of the mask matrix on the gradient. Next, we add the mask matrix $mask_e$ to G_t^k to generate the encrypted model update. Eventually, we need to calculate another mask matrix $mask_t$ for sparsification. Specifically, if the values on the same index of G_t^k and $mask_e$ are both 0, the value on the corresponding index of $mask_t$ is 0. Conversely, the value is set to 1. The local update that participant k needs to upload is $encode((G_t^k + mask_e) \bigodot mask_t)$. In the meantime, we need to calculate a residual local model update $G_{residual} = G_t^k \bigodot \neg mask_t$, which will be kept locally and accumulated in the subsequent communication rounds. Such a design can guarantee privacy while significantly reducing the upload communication overhead.

Algorithm 2: Secure Aggregation with Mask Sparsification on node k.

Input: T, dataset X, minibatch size b per node, the number of nodes N, sparsity rate R_{min}, threshold σ, α ,init parameters $w = w[0], w[1], \ldots, w[M]$

Output: G_{sparse}

for $t = 0, t \leq T, t++$ **do**
- $G_t^k \leftarrow G_{t-1}^k$;
- **for** *batch* $b \in B$ **do**
  - Sample data x from X;
  - $G_t^k \leftarrow G_t^k + 1Nb \bigtriangledown f(x; w_t)$;
- $mask_r \leftarrow$ generate a mask matrix for DH protocol;
- $loss_0 \leftarrow$ calculate loss value of local model;
- $\beta \leftarrow \frac{loss_0 - loss_k}{loss_k}$;
- $R_k \leftarrow \{R_k, R_{min}\}$;
- **for** $i = 1, i \leq M, i++$ **do**
  - Select threshold: $\sigma \leftarrow R_k of |G_t^k[i]|$;
  - $mask_{top}[i] \leftarrow |G_t^k[i]| > \sigma$;
  - **for** $j = 1, j \leq sizeof G_t^k[i], j++$ **do**
    - $mask_e[i][j] = \begin{cases} 0 & otherwise \\ mask_r[i][j] & mask_r[i][j] < \sigma \end{cases}$
    - $mask_t[i][j] = \begin{cases} 1 & otherwise \\ 0 & mask_{top}[i][j] = 0 \wedge mask_e[i][j] = 0 \end{cases}$
- $G_{sparse} \leftarrow encode((G_t^k + mask_e) \odot mask_t)$
- $G_{residual} \leftarrow G_t^k \odot \neg mask_t$
- $loss_k \leftarrow loss_0$

The participants utilize the DH protocol key as a random seed to produce a uniformly distributed mask matrix $mask_r \in [p, p + q)$. If two mask matrices correspond to the same participants and hold the same key, they are considered equal. For each participant involved in the training, the dynamic threshold can be ascertained by taking into account the total number of model gradient updates N and the sparsity rate R.

$$R = (\alpha + \beta - \frac{t}{T}) \cdot R \tag{2}$$

where α is the constant attenuation factor and β is the loss change rate of the participant. The greater the loss change, the more severe the change. t is the number of iterations and T is the specified number of training rounds. The higher the number of iterations, the smaller

the change. R is the gradient sparsity rate, where the upper limit is 1 and the lower limit is the specified minimum sparsity ratio R_{min}.

According to the threshold, we can filter out the mask matrix of the Top-k gradient update. If G is the original local gradient update, there is:

$$\forall g_i \in G, mask_{top} = |g_i| \geq \sigma \tag{3}$$

σ is the random encryption mask filtering threshold:

$$\sigma = p + \frac{k}{x} \cdot q \tag{4}$$

where x is the number of participants and k is the random mask ratio.

$mask_e$ is the random uniform distribution mask matrix filtered by σ and $mask_t$ is the mask matrix for sparse transmission. Therefore, in order to ensure the safety of the original gradient after sparsification, the gradient after sparsification is updated as:

$$G_{sparse} = encode((G + mask_e) \odot mask_t) \tag{5}$$

To make the encryption mask matrix sparse, a technique is used that involves setting a certain proportion of the matrix elements to zero. This sparseness helps to optimize communication by reducing communication costs. In the original secure aggregation framework, there was no deliberate effort to create a mask matrix with zero local values. However, such a matrix is significant in optimizing the gradient sparseness of the security aggregation framework. When certain positions in the mask matrix are set to zero, the corresponding gradient updates do not need to be transmitted in that round of communication. As a result, during the aggregation of the encrypted gradient updates at the server, there is no problem with eliminating the mask. Furthermore, since the gradient update values for these positions are not transmitted, the amount of communication data is reduced and communication efficiency improves.

4. Safety Analysis

The secure aggregation framework is a multi-party encryption computing framework. From the local gradient update encryption process for any participant, it can be inferred that as the number of participants increases, the number of encryption masks also increases, making the composition of the encryption gradient update more complex and making it more difficult for the aggregation end to judge. The local gradient updates the ground truth. It can be seen that when only two parties participate, the encryption situation is simpler and the composition of the encrypted gradient update obtained by the aggregation end is simpler. If the original gradient mask has an encryption effect, when multiple parties participate, the encryption effect of the local gradient update of the participating parties can also be guaranteed.

In a sparse security aggregation training scenario, the gradient update amplitude does not meet the threshold and the mask component of the corresponding position is zero. In this case, it will not be sent. If the encrypted gradient update meets the transmission conditions, there are several situations: the gradient update amplitude meets the threshold but the corresponding encryption mask is zero, the gradient update amplitude does not meet the threshold but the corresponding encryption mask is not zero, and the gradient update amplitude meets the threshold and the corresponding encryption mask is not zero. The three situations are described as follows:

1. **The gradient update magnitude meets the threshold but the corresponding encryption mask is zero.** This situation is equivalent to transmitting the original gradient update information. In the aggregation stage, due to the symmetry of the encryption mask matrix, if another participant happens to select the original gradient update information at the same position, the aggregation cannot infer whether the encryption

mask at this position is zero during this round of aggregation. However, as the number of iterations increases, once the corresponding position is zero, the aggregation can determine that the corresponding encryption mask is zero and reverse the previous gradient data. The true value of the original gradient at the corresponding location is exposed, although the gradient value does not represent the true data. Therefore, the aggregation knows that the mask of some positions is zero, which is not enough to expose the local data. Whether the encrypted information of the mask matrix can be deduced according to the encrypted gradient of the training process and then the original gradient information can be deduced inversely is the key to keeping local data safe.

2. **The magnitude of the gradient update does not meet the threshold but the corresponding encrypted mask component is not zero.** This situation is equivalent to transmitting random mask information. In the aggregation stage, according to the symmetry of the encryption mask matrix, there are two situations for the corresponding position: (1) another participant just selects the original gradient update information and the encryption result of the random mask at the same position, and (2) the absolute values of the encrypted values of the two participants are equal and the signs are opposite. For the first case, as long as both positions have non-zero values, the aggregation cannot directly determine whether the encryption mask is zero and or judge the value of the original gradient update. However, once the second situation, where the corresponding position is the opposite number, occurs, the encrypted mask information will be directly exposed. According to the exposed mask information, the aggregation can easily calculate the gradient of the corresponding position of the participant during the whole training process to update the original value.
3. **The gradient update magnitude meets the threshold and the corresponding encrypted mask component is not zero.** This situation is similar to the second one. Since the corresponding position is part of the random mask matrix that is not zero when at least one of the two positions contains the gradient update value, the value of the corresponding position is not the opposite and the aggregation cannot directly judge the mask condition. The multi-party training process often requires multiple rounds of iterations. As the iteration progresses, ideally, at least one of the corresponding positions will always have the gradient update range that meets the threshold. However, in the actual training process, the corresponding mask appears to be the opposite number. The situation is very likely to happen. Once a similar mask is exposed, the corresponding position will not have the corresponding encryption effect.

Three situations for sending parameters in the gradient sparse scenario expose a security problem in the original local gradient encryption during the gradient update. The local mask of zero is exposed, revealing only part of the original information, which is not the complete gradient update information. Although encryption masks can encrypt the components of local gradient updates of other participants, they prevent the aggregation from inferring the original local gradient update. However, the random encryption mask of the secure aggregation framework's symmetry results in the gradient information of some positions not being sent in this round due to the magnitude of the gradient update not meeting the threshold.

Participants do not send gradients but must send masks. It is unknown whether the gradient updates of the corresponding positions of other participants are sent during the training process. Hence, to avoid errors caused by masks, participants are randomly masked regardless of their non-zero value. The random mask information must be sent to avoid random errors after aggregation. However, when the corresponding positions of both parties do not need to send gradient update information, the values of the corresponding positions will be the opposite numbers. In this case, the value of the mask is exposed, and the subsequent training and previous encrypted gradient updates are ineffective.

The mask of the corresponding position will not change since the DH protocol executes only once in this training and involves multiple parties. Even when one party always has a value in the corresponding position, due to the absolute value of the mask, positive and negative values of equal size frequently appear in the corresponding positions. By updating the information through multiple rounds of encrypted gradients, the aggregation can easily infer the value of the mask.

Upon reviewing the training process of the original secure aggregation framework, it can be seen that the aggregation end cannot obtain the original value of the encryption mask and local gradient update because only one aggregation result is obtained and it is not safe to split them. The random mask matrix protects the original value of the local gradient update, but in turn, the local gradient update also protects the mask value from being obtained by the aggregation. The composition of the two encryption results provides high security.

Therefore, we use the dynamic sparsity rate method based on the training loss of each participant, which helps to achieve faster convergence. At the same time, because the sparsity rate of each participant is different, the index of the Top-k parameter will not have a direct impact on the encryption location. A dynamic Top-k gradient parameter helps to protect the original Top-k gradient update. Even if the local Top-k parameters are obtained, the aggregate does not know the total number of Top-k parameters, and the encrypted gradient update with the non-zero corresponding positions cannot determine which one belongs to the original gradient update. Even in extreme cases, the index and encryption location of Top-k do not overlap at all, but because the aggregation end cannot determine the specific number of Top-k parameters, the original Top-k gradient update cannot be inferred.

5. Experiments

To demonstrate the utility of our proposed approaches, we conducted a series of image classification experiments on the MNIST, Fashion-MNIST [30], and CIFAR-10 datasets. The MNIST dataset consists of 60,000 training samples and 10,000 test samples. Each sample contains 28 $\times$ 28-pixel grayscale handwritten digits. The FMINST dataset is similar to the MNIST dataset, except that the handwritten digits are replaced with commodity images. The CIFAR-10 dataset contains 60,000 natural images in ten object classes, which consist of 50,000 training pictures and 10,000 test pictures. We use the conventional FL algorithm and the improved algorithm proposed in this paper to train the MNIST-MLP, MNIST-CNN, CIFAR10-CNN, CIFAR10-VGG16, and other models. The goal of our experiments is to demonstrate that compared to other FL algorithms with no compression, our proposed method can achieve high computational and communication efficiency without affecting accuracy.

Following the methodology of [2], we trained a shared model with 100 total clients, 10 of whom were selected randomly in each round. The number of local training iterations was 5 and the training batch size was 50. We used the sample allocation matrix to simulate non-IID (independent and identically distributed) training data in the real world and provide different clients with an unbalanced sample from each class. Specifically, in order to simulate the distributed characteristics of the dataset in the real world, the complete training dataset needed to first be segmented and allocated to the client in the experiment.

5.1. Selection of Sparsity Rate and Attenuation Factor

Based on FedAvg, under the IID setting and the experimental configuration, the gradient update operation uploaded by the customer during each iteration was sparsed with different sparsity rates, and experiments with sparsity rates of s = 0.1, 0.01, and 0.001 were carried out, respectively. The experimental results are shown in Figure 1.

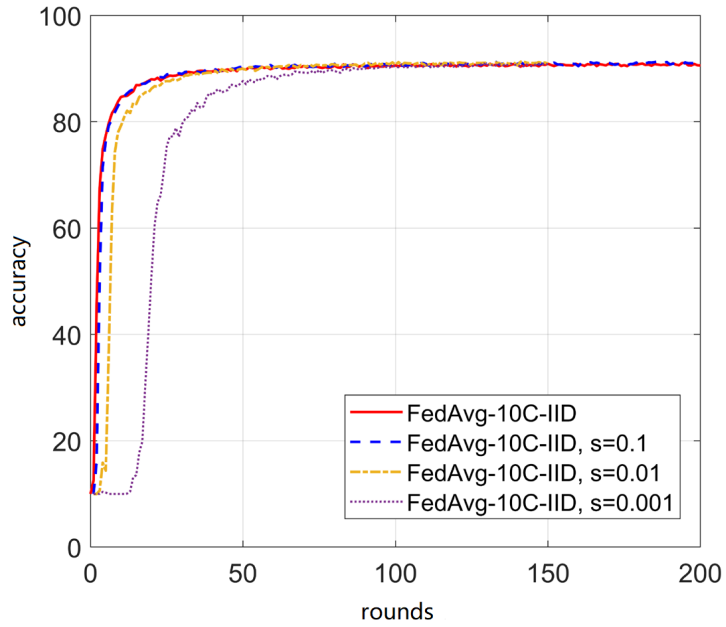

Figure 1. The accuracy of the aggregation model updated with gradient sparsification with different sparsity rates.

It can be seen that when the sparsity rate was 0.1, sparsity had almost no effect on the convergence speed of the aggregation model and the final prediction accuracy. When the sparsity rate was further increased to 0.01 and 0.001, the early iterations of the aggregation model were slowed down. To a certain extent, the performance of the aggregation model quickly improved after several rounds of iterations. Although the convergence speed itself was somewhat lower than that of non-sparseness, the predictive ability of the final model was hardly lost when it converged. The introduction of the algorithm had a negative impact on the performance of the algorithm, which was mainly reflected in a reduction in the convergence speed, but this loss was almost negligible compared to the reduction in communication costs caused by sparseness, as shown in Figure 1. At a sparsity rate of 0.001, the model took approximately 4 times the number of rounds to converge to the optimal level compared to non-sparseness, but the amount of communication required for each round was reduced to only one-thousandth of the original. In addition, the communication costs were reduced by hundreds of times. In the same network environment, the time and communication costs required to complete a round of sparse updates were much smaller. Therefore, from the perspective of time, sparseness can speed up the acquisition of models.

When the sparsity increased, s gradually decreased, and it was possible to remove important values from the local model update during the sparsification procedure. In particular, in the early stage of federated training, the changes in the model updates were relatively large, resulting in many large values. Therefore, when the server aggregated the model updates submitted by the participants, some important gradient information was missing, causing the gradient direction to deviate from the expected path and the model performance to slowly improve. When the training of the model reached a certain stage, the gradient itself had a lot of relatively small values. Therefore, gradient sparsification barely resulted in a loss of important information, which did not affect the improvement of the model performance.

For the non-IID dataset, sparsity was still effective. Take a sparsity rate of $s = 0.001$ as an example. As shown in Figure 2, through the observation of the loss curve, it can be seen that in some cases, after sparse convergence, the loss curve decreased more smoothly than after non-sparse convergence. It is generally believed that the loss curve decreases first and then increases because the model training is sufficient and further training will result in the overfitting of the model. The goal of sparsification is to update only the most important parameters, which allows each federation participant to transmit only those updates. This approach avoids parts that may have been fitted in the local training model of each client from being aggregated into the server's aggregation model, thereby avoiding the overfitting of the model, so that the generalization ability of the aggregation model can be improved.

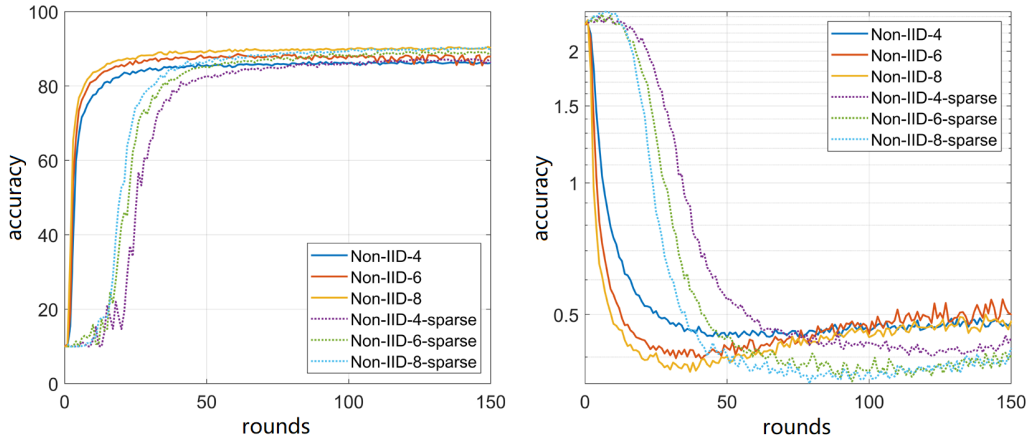

Figure 2. Under non-IID distribution, with a sparsity rate of $s = 0.001$, the learning curve of the aggregation model under the sparse updates.

We assume that the constant attenuation factor β is 0.2, 0.5, and 0.8 respectively, and the lower limit of the sparsity rate s_{min} is 0.01. In the iterative process, under these three sparsity rate settings, tests were carried out under the three conditions of non-IID-4, non-IID-6, and non-IID-8, where non-IID-n (n = 1, 2, . . . 10) represents a sample with only n types of tags in the client. The results are shown in Figure 3. The solid line represents the experimental results of the FedAvg algorithm. The long dotted line (- spark) indicates the experimental results using the conventional sparsification method based on FedAvg and the short dotted line (- layerspares) indicates the experimental results using the THGS method proposed in this paper based on FedAvg.

It can be seen that under the three constant attenuation factors, the time-varying hierarchical training effect was better than the conventional sparsity update experimental results. With the improvement of β, the effect of the algorithm continued to approach the non-sparsification algorithm, and when β was 0.8, the loss caused by sparsification could almost be ignored. There were no differences between the sparsification and non-sparsification updates on the learning curve. Based on the experimental results, we can see the optimization effect of the THGS method.

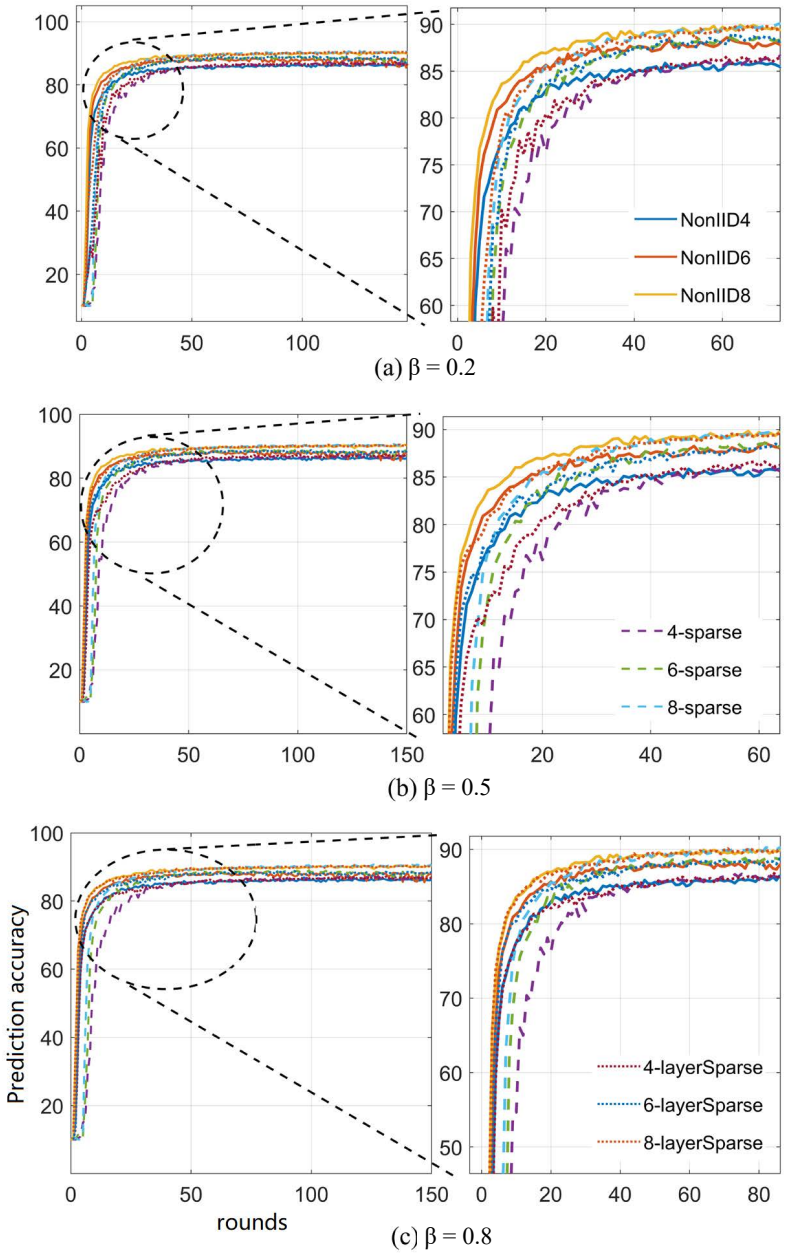

Figure 3. Experimental results under non-IID-4/6/8 distribution with different β.

5.2. Communication Costs

Assuming that the total parameters of the model are m, only the space required to store its gradient vector was calculated, and the subsequent operations such as coding and compression were not considered. Assuming that each parameter is stored in a double-precision floating point, the space required for storing a non-sparse gradient vector update is $m \cdot 64bit$.

For a sparse gradient update vector with a sparsity rate of s, since there are a large number of 0 elements in the vector, it is not necessary to store these vector vacancies. Only the position index and corresponding value of the non-zero elements in the vector need to be stored. Therefore, the space required to store them is:

$$m \cdot s \cdot 64bit + m \cdot s \cdot 32bit = ms \cdot 96bit \tag{6}$$

where $64bit$ represents the storage space of a double-precision floating point and $32bit$ represents the position index of the non-zero elements in the storage sparse vector in

the entire vector. Then, for the overall process of FL, the total communication overhead required to train a model is:

$$c_{percent} = n_{percent} \cdot ((C \cdot K) \cdot (c_{up} + c_{down})) \tag{7}$$

where $n_{percent}$ is the number of aggregation rounds required when the prediction accuracy of the model reaches the target convergence accuracy, $C \cdot K$ is the number of clients selected in each aggregation iteration, and c_{up} and c_{down} represent the communication costs required for a single client to upload and download a gradient update, respectively:

$$\begin{cases} c_{up} = \begin{cases} m \cdot s \cdot 96bit, & if sparse \\ m \cdot 64bit, & else \end{cases} \\ c_{down} = m \cdot 64bit \end{cases} \tag{8}$$

For the models used in the experiment, the parameter volumes of different models could be obtained according to the above calculation method. The specific data are shown in Table 1.

Table 1. Different model parameter sizes and update volumes.

	MNINST		**Fashion-MNIST**		**CIFAR-10**	
	MLP	**CNN**	**MLP**	**CNN**	**MLP**	**VGG16**
parameter size	159,010	582,026	159,010	582,026	5,852,170	14,728,266
update volume	1.2 M	4.44 M	1.2 M	4.44 M	44.6 M	112 M

According to the number of aggregation rounds required for convergence, the communication costs required for the different algorithms to complete an FL under different experimental conditions could be calculated. For simplicity, the upload communication costs required to make the aggregation model reach 95% of the final average convergence accuracy under the non-IID setting were calculated and the data are shown in Table 2.

Table 2. Under non-IID distribution, the upload communication costs required to reach 95% accuracy when the final average convergence is achieved.

	MNINST		**Fashion-MNIST**		**CIFAR-10**	
	MLP	**CNN**	**MLP**	**CNN**	**MLP**	**VGG16**
FedAvg	840 M	1332 M	432 M	4.68 G	94.5 G	156 G
	×13.6	×6.11	×7	×19.8	×34	×24.6
FedProx	444 M	1154 M	552 M	5.78 G	77.5 G	128 G
	×7	×5.3	×9	×24.5	×28	×20.2
Ours	61.8 M	218 M	61 M	242 M	2.77 G	6.33 G

It can be seen that the compression of the uploaded data mainly came from the gradient and mask sparsification proposed in this paper, which increased compression by 5.3 to 34 times. Considering that in an actual scenario the upload bandwidth of a device is generally far less than the download bandwidth, the algorithm in this paper can reduce the upload communication overhead by dozens of times. This optimization is considerable. Since FedAvg and FedProx upload the whole model update during each communication round, the communication overhead of each participant in each round is considerably high. Moreover, FedAvg and FedProx require more rounds to converge compared to our proposed framework. Therefore, FedAvg and FedProx always require more communication overhead to achieve the targeted goal.

5.3. Comparison with State-of-the-Art Methods

In this section, we compare our proposed framework with several state-of-the-art methods to illustrate the effectiveness of our work. Figure 4 shows the prediction accuracy of the VGG16 network on the CIFAR10 dataset. The results show that our proposed framework can achieve higher accuracy than DGC and STC. In addition, the convergence speed and stability of our method are superior to those of DGC and STC because in the early stage of federated training, the changes in model updates are relatively large, resulting in many large values. DGC and STC always adopt the same small sparsity ratio, thereby missing some important gradient information. Therefore, the convergence speed will be slow and the stability of the model will be poor in the early training stage. Since our proposed framework adopts a time-varying hierarchical gradient sparsification strategy, the important gradient information can be retained in the early training stage, which barely affects the model performance and stability.

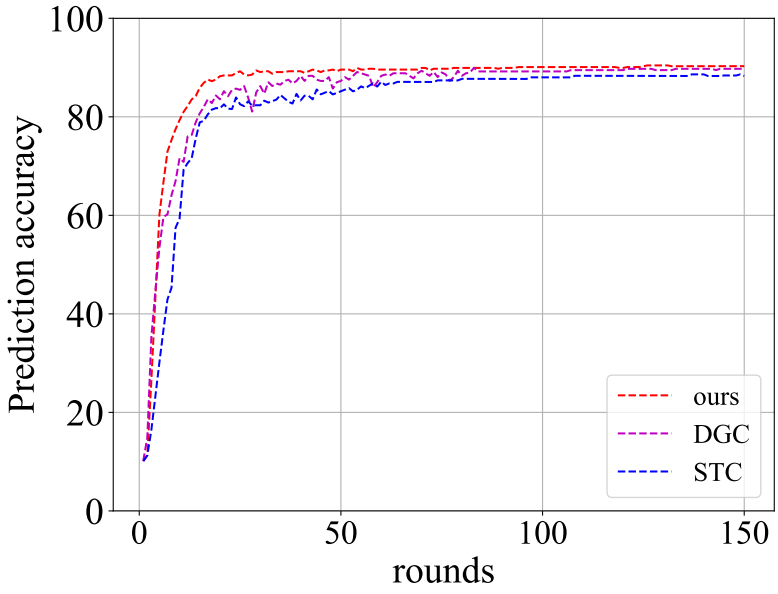

Figure 4. Comparison with state-the-of-art methods.

6. Conclusions

In this paper, we present our research findings on the application of sparsity to secure aggregation in federated learning. Our main contributions are:

- We propose an efficient and secure FL framework that can significantly reduce the communication costs of a single communication while ensuring privacy.
- Our proposed framework results in fewer model performance losses than the traditional sparsity method.
- Experiments conducted under various non-IID experimental settings demonstrate that the proposed algorithm can reduce the upload communication costs to about 2.9% to 18.9% of the conventional FL algorithm when the sparsity rate is 0.01.

Overall, our research demonstrates the potential benefits of incorporating sparsity into federated learning and presents a promising approach for achieving better communication efficiency while maintaining data privacy and model performance.

Future work could involve adding gradient correction, batch-normalized updates, and local gradients to the sparse gradient update process to maintain model accuracy after high-ratio sparsification. Additionally, adaptive sparsity could be used to automatically control the trade-off between optimal communication and computation. In terms of security aggregation, a DH key exchange before each round of training may increase the training time. Although the sparse mask matrix is conducive to increasing the training speed,

the extra time needed for the DH key exchange may affect the overall communication optimization. This aspect needs to be further studied.

Author Contributions: Conceptualization, T.L.; methodology, Z.W.; software, H.H.; investigation, W.S.; data curation, L.L.; writing—original draft preparation, R.A.; writing—review and editing, C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by NSFC grant number 62176205.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The authors conduct classification experiments using three public datasets, i.e., MNIST (http://yann.lecun.com/exdb/mnist/ (accessed on 24 November 2022)), Fashion-MNIST (https://github.com/zalandoresearch/fashion-mnist (accessed on 24 November 2022), or [30]) and CIFAR-10 (http://www.cs.toronto.edu/kriz/cifar-10-matlab.tar.gz (accessed on 24 November 2022)).

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

Abbreviations

The following abbreviations are used in this manuscript:

ML	Machine learning
FL	Federated learning
CNN	Convolutional neural networks
DP	Differential Privacy
HE	Homomorphic Encryption
SMC	Secure Multiparty Computation
SA	Secure Aggregation
DGC	Deep gradient compression
STC	Sparse compression framework
THGS	Time-varying hierarchical gradient sparsification

References

1. Bai, G.; Xi, W.; Hong, X.; Liu, X.; Yue, Y.; Zhao, S. Robust and Rotation-Equivariant Contrastive Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *early access*.
2. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the Artificial Intelligence and Statistics, PMLR, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
3. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60. [CrossRef]
4. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.A.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.* **2021**, *14*, 1–210. [CrossRef]
5. Wang, S.; Tuor, T.; Salonidis, T.; Leung, K.K.; Makaya, C.; He, T.; Chan, K. Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1205–1221. [CrossRef]
6. Huang, A. Dynamic backdoor attacks against federated learning. *arXiv* **2020**, arXiv:2011.07429.
7. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
8. Sattler, F.; Wiedemann, S.; Müller, K.R.; Samek, W. Sparse binary compression: Towards distributed deep learning with minimal communication. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; pp. 1–8.
9. Bernstein, J.; Wang, Y.X.; Azizzadenesheli, K.; Anandkumar, A. signSGD: Compressed optimisation for non-convex problems. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 560–569.
10. Bhowmick, A.; Duchi, J.C.; Freudiger, J.; Kapoor, G.; Rogers, R. Protection Against Reconstruction and Its Applications in Private Federated Learning. *arXiv* **2018**, arXiv:1812.00984.
11. Melis, L.; Song, C.; De Cristofaro, E.; Shmatikov, V. Exploiting unintended feature leakage in collaborative learning. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; pp. 691–706.
12. Dwork, C.; Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.* **2014**, *9*, 211–407. [CrossRef]

13. Acar, A.; Aksu, H.; Uluagac, A.S.; Conti, M. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–35. [CrossRef]
14. Evans, D.; Kolesnikov, V.; Rosulek, M. A pragmatic introduction to secure multi-party computation. *Found. Trends® Priv. Secur.* **2017**, *2*, 70–246. [CrossRef]
15. Halevi, S.; Lindell, Y.; Pinkas, B. Secure computation on the web: Computing without simultaneous interaction. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 132–150.
16. Leontiadis, I.; Elkhiyaoui, K.; Molva, R. Private and dynamic time-series data aggregation with trust relaxation. In Proceedings of the International Conference on Cryptology and Network Security, Heraklion, Greece, 22–24 October 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 305–320.
17. Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical secure aggregation for privacy-preserving machine learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1175–1191.
18. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Dallas, TX, USA, 30 October–3 November 2016; pp. 770–778.
19. Jiang, P.; Agrawal, G. A Linear Speedup Analysis of Distributed Deep Learning with Sparse and Quantized Communication. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; Volume 31.
20. Strom, N. Scalable distributed DNN training using commodity GPU cloud computing. In Proceedings of the Sixteenth Annual Conference of the International Speech Communication Association, Dresden, Germany, 6–10 September 2015.
21. Dryden, N.; Moon, T.; Jacobs, S.A.; Van Essen, B. Communication Quantization for Data-Parallel Training of Deep Neural Networks. In Proceedings of the 2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC), Salt Lake City, UT, USA, 14 November 2016; pp. 1–8. [CrossRef]
22. Ba, L.J.; Kiros, J.R.; Hinton, G.E. Layer Normalization. *arXiv* **2016**, arXiv:1607.06450.
23. Lin, Y.; Han, S.; Mao, H.; Wang, Y.; Dally, B. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In Proceedings of the ICLR (Poster), Vancouver, BC, Canada, 30 April–3 May 2018.
24. Han, P.; Wang, S.; Leung, K.K. Adaptive Gradient Sparsification for Efficient Federated Learning: An Online Learning Approach. In Proceedings of the 40th IEEE International Conference on Distributed Computing Systems, ICDCS 2020, Singapore, 29 November–1 December 2020; pp. 300–310. [CrossRef]
25. Qiu, X.; Fernández-Marqués, J.; de Gusmao, P.P.B.; Gao, Y.; Parcollet, T.; Lane, N.D. ZeroFL: Efficient On-Device Training for Federated Learning with Local Sparsity. In Proceedings of the Tenth International Conference on Learning Representations, ICLR, Virtual Event, 25–29 April 2022.
26. Alistarh, D.; Li, J.; Tomioka, R.; Vojnovic, M. QSGD: Randomized Quantization for Communication-Optimal Stochastic Gradient Descent. *arXiv* **2016**, arXiv:1610.02132.
27. Diffie, W.; Hellman, M.E. New directions in cryptography. *IEEE Trans. Inf. Theory* **1976**, *22*, 644–654. [CrossRef]
28. So, J.; Güler, B.; Avestimehr, A.S. Turbo-Aggregate: Breaking the Quadratic Aggregation Barrier in Secure Federated Learning. *IEEE J. Sel. Areas Inf. Theory* **2021**, *2*, 479–489. [CrossRef]
29. Kadhe, S.; Rajaraman, N.; Koyluoglu, O.O.; Ramchandran, K. FastSecAgg: Scalable Secure Aggregation for Privacy-Preserving Federated Learning. *arXiv* **2020**, arXiv:2009.11248.
30. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* **2017**, arXiv:1708.07747.