Hindawi
Scientific Programming
Volume 2022, Article ID 7465640, 9 pages
https://doi.org/10.1155/2022/7465640

Research Article

Privacy Shield: A System for Edge Computing using Asynchronous Federated Learning

Adnan Khalid,[1] **Zeeshan Aziz**,[2] **and Mohamad Syazli Fathi**[3]

[1]*Department of Computer Science and Information Technology, The University of Lahore, Lahore, Pakistan*
[2]*School of Science, Engineering and Environment Maxwell Building, University of Salford, Manchester, UK*
[3]*Universiti Teknologi Malaysia, Razak Faculty of Technology and Informatics, Jalan Sultan Yahya Petra, Kuala Lumpur, Malaysia*

Correspondence should be addressed to Adnan Khalid; adnan.khalid@gcu.edu.pk

Received 23 March 2022; Revised 27 April 2022; Accepted 7 May 2022; Published 12 July 2022

Academic Editor: Sikandar Ali

Due to increase in IoT devices, the data produced every day are also increasing rapidly. The growth in data means more processing and more computations are required without delay. This introduced us to a new horizon of the computing infrastructure, i.e., edge computing. Edge computing gained prominence as a solution to the problem of delayed transmission, processing, and response by the cloud architecture. It was further augmented with the field of artificial intelligence. It has become a topic in research with preservation of data privacy as the focal point. This paper provides Privacy Shield, a system for edge computing using asynchronous federated learning, where multiple edge nodes perform federated learning while keeping their private data hidden from one another. Contrary to the pre-existing distributed learning, the suggested system reduces the calls between the edge nodes and the main server during the training process ensuring that there is no negative impact on the accuracy of the mode. We used different values of Ω that affect the accuracy and the compression ratio. In the interval of $\Omega = [0.2, 0.4]$, the C-ratio increases and the value of Ω and the compression ratio value are directly proportional regardless of the fluctuations. We analyzed the accuracy of the model which increases along with the increase in compression ratio. Compressing the gradient communications reduces the likelihood of the data of being attacked. As the nodes train asynchronously on limited and different kinds of samples, the binary weights adjustment method was used to handle the resulting "unbalanced" learning. The MNIST and Cifar10 dataset were used for testing in both tasks, respectively.

1. Introduction

IoT plays a significant role in our day-to-day life and has greatly impacted the efficiency of the business world. IoT is responsible for generating an enormous amount of data which the companies store in their cloud-based server. This is where the necessary processing of the data takes place and effective inference models are extracted. The Internet giants collect myriads of training corpora from the client side to implement deep neural network. The indisputable extensive usage of deep learning, however, has some downsides.

Furthermore, the cloud-based approach cannot handle the bulk of data produced by the IoT devices which results in a slowdown in data processing and transmission bottleneck. This can be hazardous specifically to time-critical applications.

To combat the challenges mentioned above, we have proposed Privacy Shield, a system for edge computing using asynchronous federated learning that addresses the real-time learning of data from multiple sources without unveiling their own private information. Our method, as compared to the customary distributed learning, makes sure that the accuracy of each party's local model is impeccable. Every node trains autonomously on a local set of data. For higher accuracy of the models, models extracted by the participants are used for the purpose of optimizing parametric quantities of the global general model.

Our suggestions in this paper are itemized below:

(1) We offer an apt asynchronous federated learning system where the collaborative learning across the autonomous vertices, takes place except that there is no sharing of private information.

(2) We devise and improvise an algorithmic program, the gradient compression algorithm. It compresses the gradient communications and minimizes the possibility of interception.

(3) We resolve the problem of deterioration of performance due to asynchronous learning of the edge nodes by designing, proposing, and validating a method called binary weights adjustment method.

The remaining part of this paper is structured as follows: Section 1 gives a concise introduction of our work. Section 2 presents the already available solutions for federated learning. Section 3 shows the core structure of the proposed system, comprising two parts: gradient compression by a self-adjusting threshold and asynchronous federated machine learning using binary weights adjustment. It continues to elaborate the gradient compression by a self-adjusting threshold, and examines its test-free apparatus. It further suggests a binary weights adjustment method for asynchronous federated machine learning. Section 4 illustrates the experiment of our proposed methods. Section 5 concludes our work.

2. Related Work

Basically, federated learning is a decentralized learning approach which enables the training data to be stored on end devices where the learners' (nodes) local updates are aggregated by a shared general model. Additionally, the self-reliant nodes ensure data integrity. The distributed architecture of blockchain can aid even the federated learning global server to become decentralized in nature, thereby, eliminating any points of malfunction or failure problems. Furthermore, it can also resolve the issues involving the security of the global server [1–6].

Nonetheless, our focus here is on the traditional federated learning. Most of the successful work performed on federated learning such as the McMahen's federated average algorithm [7] and security aggregation [8], have used synchronous training. Implementation of federated learning in vehicle-to-vehicle communication [9] and in medical applications [10] also falls under the category of synchronous learning. Security in federated learning such as security aggregation [8] requires device-level operations for synchronization, so it technically falls in the category of synchronous training.

To minimize the problems of communication bottlenecks, various methods have been proposed. Gradient quantization and sparseization are one of the prominent ones. Gradient quantization significantly cuts down the communication bandwidth cost. In conventional speech applications, the 1 bit Stochastic Gradient Descent (SGD) [11] scales down the size of the gradient transmission data to attain ten times more speed while TernGrad [12] uses a 3-level gradient on CNN and QSGD.

Both of these methods clearly illustrate the convergence of quantitative training. Gradient sparseness compresses the gradient communication to reduce network pressure. What strikes initially is periodically omitting some of the gradient interactions [13]. Dryden et al. [14] chose a predefined ratio of positive and negative gradients, respectively. Chen et al. [15] came up with a mathematical model that adjusted the compression ratio all by itself in accordance with the local gradient activity and achieved 200 times the compression and 40 times the compression for the fully connected layer and for the convolution layer respectively. In addition to this, it achieved top-1 accuracy which is trivial in the ImageNet dataset. However, disregarding the information of real-time gradients and compressing communications in accordance with the predefined ratio or an interval can have a negative impact on the training process. Strom [16] made use of thresholds to achieve gradient sparseness by only sending gradients that have a greater value than the predefined fixed threshold. Gradient dropping [17] made use of a single absolute threshold value to sparse the gradient matrix and saves 99% of gradient swaps resulting in a 0.3% loss of BLEU during the machine translation procedures.

However, the tricky part is choosing the accurate threshold value. One "perfect" fixed threshold may be an ideal choice for a particular scenario, but the very same threshold may be the worst choice for another. Furthermore, as the selection of a threshold is time-consuming, it is inefficient for scenarios that require swift response.

At the moment, the Lazily Aggregated Gradient (LAG) [18] seems appropriate as it is robust for gradient calculation and for partial gradient communications in order to decrease the server pressure and reduce the communication bandwidth. The fundamental principle is to detect the gradual change in gradient and then to compress it. Although it is remarkable, it confines the problem of optimization to be obvious and Lipschitz smooth.

Moreover, defining gradient thresholds adversely affects the scalability of the asynchronous federated learning framework. Implementation of synchronous learning becomes an arduous task if the learning object is pushed to the edge nodes. Few research studies which have used the asynchronous learning method, have done so by using it as a regularization method [19], and they lack in detailed methodical research. Furthermore, few research studies [6, 20–22] has addressed this issue and explained different methodologies in an effective way.

3. Privacy Shield: A System for Edge Computing using Asynchronous Federated Learning

Federated learning is an additional approach to machine learning where a shared general predictive model enables the processing nodes to perform collaborative learning while keeping the training data at the end devices, thus safeguarding the private data. A classic mechanism of federated learning involves a main server and numerous dispersed edge nodes where every node locally learns from its respective sensitive data as shown in Figure 1. The function of the server is collect gradients transmitted to it by every independent learning node. It then updates the criteria of the predictive model as per the optimization algorithm, and maintains the global parameters.

The participants then download the updated guidelines from the main server, override their local model parameters, and go on to the next iteration.

3.1. Gradient Compression by a Self-Adjusting Threshold. Gradient compression minimizes the number of calls made by an edge node to the main server. As discussed above, the gradient contains redundant information. The increase in the number of edge nodes raises the communication cost exponentially. Moreover, omitting redundant exchanges can lessen the communication load the main server is under. Furthermore, the gradient data is indicative of the sample information held secretly by the nodes. Intruders can infer illustrative target node data by combining the trained global model with the collected effective gradient information. Gradient compression gives a boost to the performance of federated learning. It unburdens the network load as well as reduces the likelihood of sample privacy breaches. In the prior work [13–17], be it only including the communication compression ratio or compressing gradient communications based on a predefined threshold, there are many drawbacks.

3.1.1. Mathematical Derivation

Table 1 contains a symbolic representation of the formulas used.

The main server communicates with E learning edge node to update of the parameters of the global model.

In the ith iteration, the main server transmits the current model $\omega^{(i-1)}$ to all the nodes where each learning edge node $e \in E$ calculates $\gamma_{e-1}(\omega^i)$ and sends it to the main server. The server then gathers the gradients from all the participating nodes and updates the parameters of the model by recapitulating the gradient descent algorithm.

$$\omega^i = \omega^{i-1} - \beta y_E^{i-1}, \tag{1}$$

where β is the learning rate and γE^{i-1} is the aggregation gradient, which represents a round of variation of the model.

By using the Lazily Aggregated Gradient (LAG) [12], we reduced the communication bandwidth strain, i.e., the main server communicates with the nodes which satisfy the threshold called the "Hard Work" nodes (E_H) and not with the "Lazy" ones (E_L) that do not. Thus, the total number of nodes in the network are

$$E = E_L + E_H. \tag{2}$$

The lazy nodes accumulate the gradient locally till it surpasses the threshold. So, γ_E^{i-1} becomes

$$y_E^{i-1} = y_{E_L}^{i-1} + y_{E_H}^{i-1}. \tag{3}$$

Let E_L satisfy the inequality given below:

$$\frac{\left\| y_{E_L}^{i} \right\|^2}{e_L} \leq \frac{\left\| y_{E_H}^{i-1} \right\|^2}{e}, \tag{4}$$

where e is the total number of nodes in set E and e_L is the total number of nodes in set E_L.

By putting value of γ_E^{i-1} from (1) in (3), we get

$$\left\| y_{E_L}^{i-1} \right\|^2 \leq \frac{E}{\beta^2 E} \left\| \omega^i - \omega^{\mathrm{i}-1} \right\|^2, \tag{5}$$

where $||\gamma_{EL}^{i-1}||^2 = ||\Sigma_{e \in EL}\ \gamma\ (\omega^i)||^2$.

As per the Inequality of Arithmetic and Geometric Means, $||\gamma_{EL}^{i-1}||^2$ satisfies equation (5):

$$\left\| y_{E_L}^{i-1} \right\|^2 \leq E_L \Sigma_{e \in E_L} \left\| y_e\left(\omega^i\right) \right\|^2. \tag{6}$$

If the node $e \in E_L$ satisfies the following conditions, equation (3) will be applicable:

$$\left\| y_e\left(\omega^i\right) \right\|^2 \leq \frac{1}{\beta_{e_L \mathrm{e}}^2} \left\| \omega^i - \omega^{i-1} \right\|^2. \tag{7}$$

We simplify equation (6) by introducing a proportional coefficient Ω to represent the total number of nodes of the set E_L, that is, $e_L = \Omega_e$. This is because we cannot identify the total number of $e \in E_L$ beforehand.

$$\left\| y_e\left(\omega^i\right) \right\|^2 \leq \frac{1}{\beta^2 \Omega e_L \mathrm{e}} \left\| \omega^i - \omega^{i-1} \right\|^2. \tag{8}$$

Because of the smooth changes in parameters during the learning process, $\omega^i - \omega^{i-1}$ is estimated as

$$\omega^i - \omega^{\mathrm{i}-1} \approx \sum_{Z=1}^{z} \theta_z \left(\omega^{i-z} - \omega^{\mathrm{i}-z-1} \right), \tag{9}$$

where θ_z and Z are constant coefficients, and we choose $\theta_z = 1/Z$.

By putting the value of $\omega^i - \omega^{i-1}$ from (8) in (7), we get the required expression:

$$\left\| y_e\left(\omega^i\right) \right\|^2 \leq \frac{1}{\beta^2 \Omega \mathrm{e}^2} \left\| \sum_{z=1}^{z} \theta_z \cdot \left(\omega^{\mathrm{i}-z} - \omega^{i-z-1} \right) \right\|^2. \tag{10}$$

The nodes conduct the self-test operation at the end of every iteration. Only those nodes will communicate with the main server which do not satisfy (9). The ones that satisfy it will accumulate the gradient locally and go on to next iteration of learning.

3.1.2. Gradient Compression without Examination. To mitigate the local computation time, we have included an apparatus without examination too.

Let I be a set of total epochs, i.e.,

$$I = \{I_1, I_2, \ldots \mathrm{I_n}\}, \tag{11}$$

where the kth variable $I\ k \in I$ indicates the possibility of nodes that directly communicate with the main server after the *kth* iteration without undergoing the gradient check. If and only if $I_k \geq I_T$, the node can pass up the self-test, where IT is the default probability threshold.

3.2. Asynchronous Federated Machine Learning by Edge Network. The proposed system centers around unrestrained network edge nodes dealing with heterogeneous sources of data and learn asynchronously. This can result in different computing power and time for training of a particular task

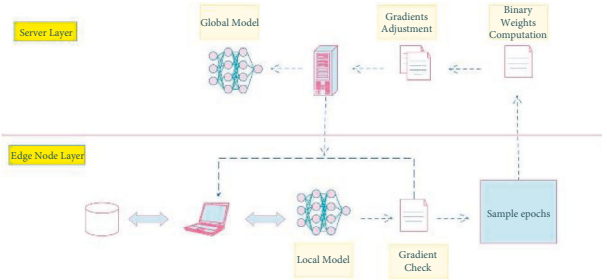

Figure 1: Privacy shield, a system for edge computing using asynchronous federated learning.

Table 1: Symbols and their meanings used in the forthcoming formulas.

Symbol	Meaning
E	The set of edge nodes
S	Total number of samples owned by every node
S_j	Node j samples
γ_S^j	Node j sample weight
γ_P^j	Parameter weight of node j
ω^i	The parameter of the i_{th} iteration of the main server
$\Omega_e\ (\omega^i)$	The i_{th} round gradient calculated by node e as per the i_{th} iteration parameter
Ω_E^i	The sum of the i_{th} round gradient of all nodes in t E

by each node. We propose binary weights adjustment method to solve these learning problems.

3.2.1. Asynchronous Federated Machine Learning without Binary Weights Adjustment. The nodes performing federated learning asynchronously have diverse and heterogeneous learning samples and have different learning state.

In Figure 2, the horizontal line represents the current time. The total iterations for learning are same for all the participating nodes. Each node takes its time for training depending on the external factors.

3.2.2. Binary Weights Adjustment for Asynchronous Federated Learning. The two weights are as follows: the sample weight and the parameter weight.

The sample owned by a specific edge node to the total number of samples of all nodes gives the sample weight.

Let E be the set containing all the nodes.

$E = \{e_1, e_2, \ldots, e_n\}$ and

S be the total number of samples.

To get the sample weight of an edge e_i, we divide its number of samples S_i by S, i.e.,

$$y_s^j = \frac{s_i}{s}, \tag{12}$$

where $\gamma^j S$ is the sample weight of node e_i and $S = \sum_{j=0}^{n} S_j$ is the total sample number of n nodes.

The *parameter weight* is determined the temporal gap between reception of parameters and transmission of gradients by a node.

This process of downloading and uploading is, in a way, a recurrent pattern optimize the global parameters.

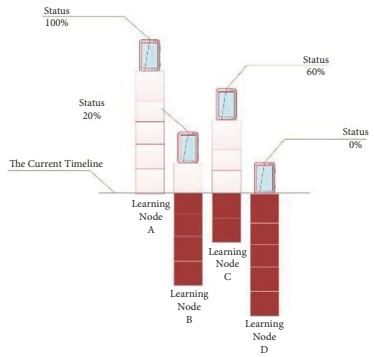

Figure 2: Asynchronous federated learning.

The gradients being uploaded have a certain "staleness" to them. The parameters of the nodes are outdated and inaccurate. Staleness is the processing power of the nodes, i.e.,

$$\Lambda_{\text{staleness}} = I_{\text{upload}} - I_{\text{download}}. \tag{13}$$

As Figure 3 suggests, staleness is

$$\Lambda_{\text{staleness}} = t + \tau - t = \tau. \tag{14}$$

We selected an exponential function with base less than 1 as the attenuation function of the parameter weights so that the nodes possessing higher staleness have smaller parameter weights and their process of weight attenuation becomes somewhat levelled.where $\Lambda_{\text{staleness}}^i$ is the staleness of node i and

$$y_P^i = \beta^{\left(1/n\Lambda_{\text{staleness}}^i\right)-1}. \tag{15}$$

β tells the speed of attenuation.

We want the exponential value and the parameter weight to be inversely proportional when the former is positive, reducing the interval of β to [0, 1].

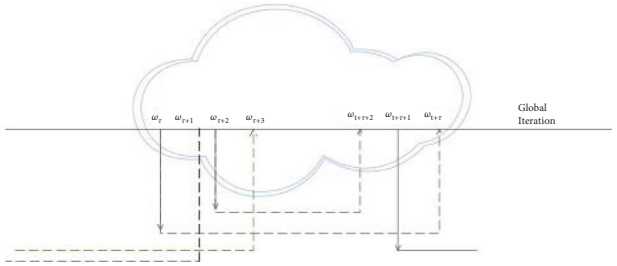

Figure 3: Parameter staleness.

Obviously, larger the β in the interval [0, 1], the more eventual the decline of the result of the calculation. For this reason, we chose $\beta = 0.9$.

The binary weights adjustment formula is given by

$$\omega\prime = \omega^* y_S^i * y_P^i, \tag{16}$$

where ω is the original parameter of model and ω' is the adjusted parameter of model.

Figure 4 clearly shows the asynchronous federated learning in the proposed system. It is an iterative process where prior the optimization of the global model occurs, and the gradient uploaded to the main server by an edge node undergoes the binary weights adjustment. The adjusted gradients then update the global parameters in accordance with the specific optimization algorithm after which the nodes override their local parameters by downloading the updated parameters.

4. Experiment

4.1. Experimental Setup. We conducted our study in an environment thusly controlled. We chose a GPU server with a high processing power acted as the main server, several personal computers, each with different computing power, played the role of learning network edges and each performed federated learning and stored data independently and locally. The process of uploading gradients and downloading updated parameters was based on the Thrift framework.

Each node's data took a fraction of the total data, i.e., 0.2% and trained its neural network model on its own private data. Due to different processing time taken by every computer, we added breaks after each epoch.

An administrative node controlled the federated learning systems of each learning nodes.

Initially, the administrator node produces a shuffled series of order as per the total number of nodes (such as 2, 3, 9, . . ., 6 in Figure 5), and produced the corresponding random series of interval (as 15.70, 45.96, 62.47, . . ., 325.12 in Figure 5). Order is the starting order of the nodes while the interval is the time difference between two adjoining epochs. After initialization, the administrator node started the selected learning node in accordance with the generated order: interval sequence pair.

4.2. Experimental Parameters. We chose three experimental indices for evaluation: compression ratio (*C*-ratio), accuracy (Accr), and compression balance index (CBI).

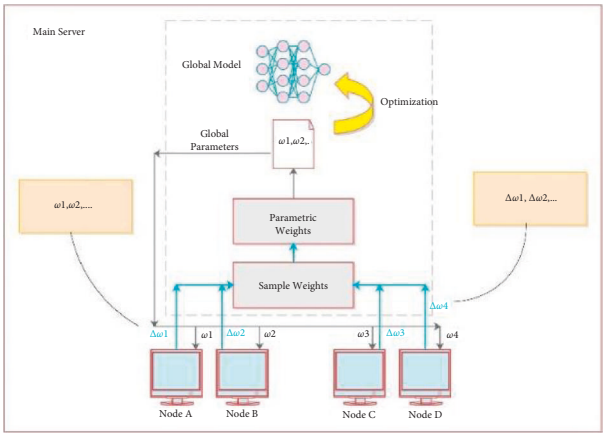

Figure 4: A detailed view of the asynchronous federal learning in the proposed system.

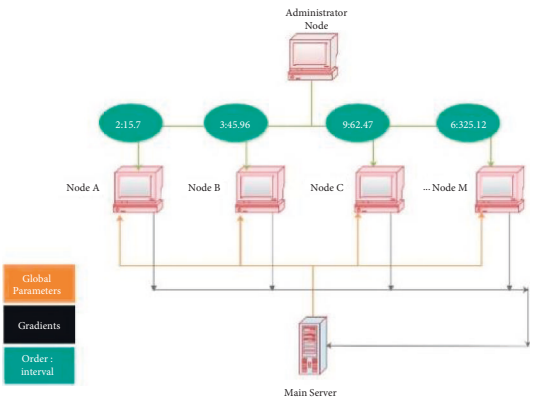

Figure 5: Experiment setup.

Accr is the measure of the performance of classification performed by the model, which is described below:

$$\frac{\text{The number of samples classified correctly}}{\text{The number of total samples}} * 100. \tag{17}$$

C-ratio is the measure of the gradient compression. So, the smaller the *C*-ratio value is, the higher is the measure of compression, described below:

$$\frac{\text{Number of calls after compression}}{\text{Number of calls before compression}} * 100. \tag{18}$$

Accr decreases with the decrease in *C*-ratio making the decision hard to make. So, CBI represents the details of performance of gradient compression, described below:

$$\text{CBI} = c_1 * \text{ACCR} + c_2 * (1 - C - \text{Ratio}), \tag{19}$$

where c_1 and c_2 are adjustable parameters to determine the precedence of Accr and C-ratio, $c_1 + c_2 = 1, c_1, c_2 > 0$.

If the precedence of Accr is greater than the C-ratio, then $c_1 > c_2$, else, $c_1 < c_2$. If both have the same precedence, then, $c_1 = c_2$. The performance of the gradient compression betters if the CBI value is high.

Table 2: Measure of compression and accuracy for varying values of Ω.

Ω	Average communication times after compression	Accr for training set (%)	Accr for test set (%)	*C*-ratio (%)
0.2	30	93.68	91.98	6.33
0.4	428	95.4	92.4	71.3
0.6	540	95.3	92.5	90
0.8	566	95.34	92.52	94.3
0.10	580	95.46	92.52	96.7
0.12	585	95.44	92.44	97.5
0.14	585	95.45	92.4	98.83
0.16	593	95.31	92.35	98.83
0.18	594	95.41	92.42	99
1	600	95.41	92.41	100

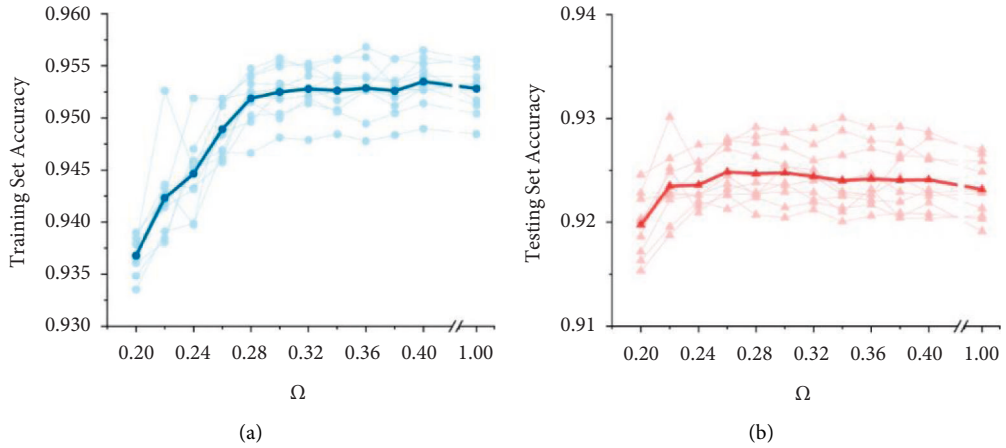

Figure 6: (a) Training set accuracy with varying values of Ω. (b) Testing set accuracy with varying values of Ω.

4.3. Experiment for Gradient Compression by a Self-Adjusting Threshold. We judged the performance of gradient compression of our system by using a handwritten image dataset, MNIST, with 60,000 samples for training and 10,000 samples for testing. We further normalized images to a 32×32 format for the purpose of making the handwritten number appear at the center of the image.

We used three training edge nodes. The model used was a three-layered multiperceptron, and the number of neurons in each layer was 256 in the input layer, 256 in the hidden layer, and 10 in the output layer. The dataset was spliced equally into three portions, and the learning nodes arbitrarily got one of them. In each conducted experiment, the nodes were given new data to work with. We made sure to test different hyperparameters Ω and averaged the end results.

We did not make noticeable changes neither to the structure of the model nor to the algorithms of optimization. Consequently, when we compare the results of training set and the test set, we can see that overfitting occurs in the model. Furthermore, the three indices for evaluation were used to compare the performance of each method only and disregarded the strengths and weaknesses of the model. Our system can be enhanced by adjusting the model structure to resolve a variety of other learning.

4.3.1. Analysis of Gradient Compression Experiment in Terms of Accr and C-ratio. Table 2 contains different values of Ω that affect the accuracy and the compression ratio. Apparently, in the interval of $\Omega = [0.2, 0.4]$, the C-ratio increases noticeably, whereas during the interval $\Omega = [0.4, 1]$, it does not increase that much.

The value of Ω and the compression ratio value are directly proportional. Regardless of the fluctuations, it is clear that the accuracy of the model increases along with the increase in compression ratio as shown in Figure 6, i.e., the bold lines clearly suggest that in intervals $\Omega = [0.14, 1]$ and $\Omega = [0.14, 1]$ of Figures 6(a) and 6(b), respectively, that the gradient compression has negligible effect on the accuracy. Although, irregularities do exist in the result but the reason for this is that the model has achieved maximum learning. In this case, the redundant gradient communications are certainly unhelpful for such a model. So gradient compressions ensure efficiency of performance, i.e., *C*-ratio increases as Ω increases because both of them are directly proportional as shown in Figure 7.

4.3.2. Analysis of Gradient Compression Experiment in terms of CBI. We computed the value of CBI in the interval $\Omega = [0.2, 0.4]$. The optimal value of Ω in our experiment turned out to be $\Omega = 0.2$.

We compared the performance of gradient compression in Privacy Shield with LAG algorithm [18] as shown in Table 3.

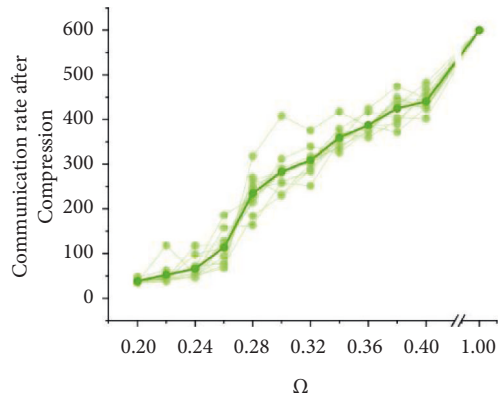

Figure 7: Communication times for different values of Ω.

Table 3: Performance of the proposed system and LAG for different values of Ω.

Compression algorithm Ω	Privacy shield		LAG
	0.2	0.22	
Accr (training set)	93.68%	94.23%	89.9%
Accr (test set)	91.98%	92.35%	89.13%
C-ratio	6.33%	8.77%	5.11%
CBI ($c_1 = 0.8$, $c_2 = 0.12$)	0.9333	0.9206	0.9274
CBI ($c_1 = 0.10$, $c_2 = 0.10$)	0.9325	0.9226	0.922
CBI ($c_1 = 0.12$, $c_2 = 0.8$)	0.9316	0.9247	0.9167

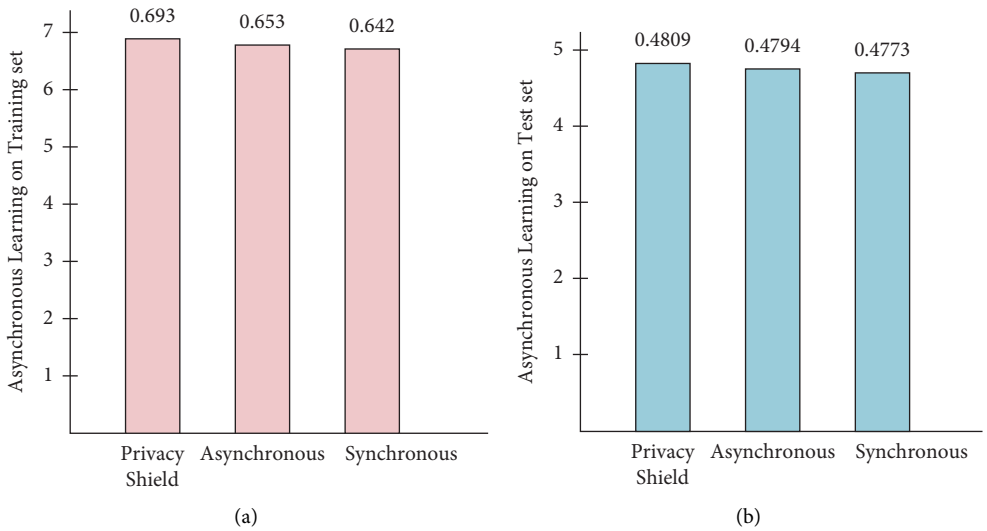

Figure 8: (a) Asynchronous federated learning performance on the training set. (b) Asynchronous federated learning performance on the test set.

4.4. Experiment for Asynchronous Federated Learning. The dataset used for the asynchronous learning was Cifar10 which consists of 50,000 training 32×32 color images and 10,000 32×32 color test images of 10 classes. We sectioned the training set into 500 parts. The learning nodes arbitrarily chose one of the parts, before the start of every experiment, as their local learning data. The total number of nodes was 10.

We used a five-layered convolutional network model the code of which is on the official website of Tensorflow.

In asynchronous federated learning trained network models on the edges are not as stable as the ones trained by the main server. So, we reduced the number of epochs for training to 500 and added breaks at the end of every learning epoch. But accuracy of the model suffered due to this reduction which can be improved by optimization or by using other means. Keep in mind that the experiment needs to be configured according to the specific problems.

The values of the hyperparameter in this experiment are arbitrary. We selected ten distinct sets of hyperparameters in

a randomized manner for comparing the synchronous learning, asynchronous learning, and Privacy Shield on the training set, i.e., no breaks during training, breaks during training, and breaks during training with the binary weights adjustment.

The bar graphs in Figures 8(a) and 8(b) reveal the performance of the ten chosen nodes under five different hyperparameter sets for both the training set and the test set.

5. Summary

This paper proposes a system, Privacy Shield, a system for edge computing using asynchronous federated learning, for the purpose of learning data from multiple sources while ensuring privacy protection. The core modules of the proposed system are: gradient compression by a self-adjusting threshold and asynchronous federated machine learning using binary weights adjustment. The former is responsible for reducing the gradient communications according to a fixed threshold. We used the handwritten image dataset known as MNIST for testing the performance of gradient compression of our proposed system. We used different values of Ω that affect the accuracy and the compression ratio. In the interval of $\Omega = [0.2, 0.4]$, the C-ratio increases and the value of Ω and the compression ratio value are directly proportional regardless of the fluctuations. We analyzed the accuracy of the model which increases along with the increase in compression ratio. Compressing the gradient communications reduces the likelihood of the data of being attacked. As the nodes train asynchronously on limited and different kinds of samples, the binary weights adjustment method was used to handle the resulting "unbalanced" learning. The dataset used for the asynchronous learning was Cifar10 and the efficiency of asynchronous federated learning reveals some major circumstances, like as trained network models on the edges are not as stable as the ones trained by the main server. So, we reduced the number of epochs for training to 500 and added breaks at the end of every learning epoch. However, accuracy of the model suffered due to this reduction which can be improved by optimization [23].

Data Availability

The data used in this research will be available upon request from the corresponding author.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

References

[1] Y. Kim and C. Hong, "Blockchain-based node-aware dynamic weighting methods for improving federated learning performance," in *Proceedings of the 20th Asia-Pacific Network Operations and Management Symposium*, pp. 1–4, Matsue, Japan, September 2019.

[2] D. Conway-Jones, T. Tuor, S. Wang, and K. K. Leung, "Demonstration of Federated Learning in a Resource-Constrained Networked Environment," in *Proceedings of the IEEE International Conference on Smart Computing*, pp. 484–486, Washington, DC, USA, June2019.

[3] U. Majeed and C. Hoong, "Federated Learning via MECenabled Blockchain Network," in *Proceedings of the 20th AsiaPacific Network Operations And Management Symposium*, pp. 1–4, Matsue, Japan, September 2019.

[4] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: reducing the communication bandwidth for distributed training," 2017, https://arxiv.org/abs/1712.01887.

[5] J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, p. 1, 2019.

[6] M. Arif and T. Mahmood, "Cloud computing and its environmental effects international journal of grid and distributed computing cloud computing and its environmental effects," vol. 8, no. 1, pp. 279–286, 2015.

[7] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: state-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416–464, 2018.

[8] K. Bonawitz and V. B. A. H. B. S. D. A. K. Ivanov, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, Dallas Texas, USA, October 2017.

[9] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultrareliable low-latency V2V communications," in *Proceedings of the GLOBECOM IEEE Global Communications Conference*, Abu Dhabi, United Arab Emirates, December2018.

[10] T. S. Brisimi and R. T. A. I. C. W. Chen, "Federated learning of predictive models from federated Electronic Health Records," *International Journal of Medical Informatics*, vol. 112, no. 1, pp. 59–67, 2018.

[11] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit Stochastic Gradient Descent and its Application to Data-Parallel Distributed Training of Speech DNNs," in *Proceedings of the Interspeech 2014, 15th Annual Conference of the International Speech Communication Association*, pp. 1058–1062, Singapore, September 2014.

[12] W. Wen, C. Xu, F. Yan et al., "Terngrad: ternary gradients to reduce communication in distributed deep learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[13] S. Wang, T. Tuor, T. Salonidis et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, 1221 pages, 2019.

[14] N. Dryden, T. Moon, S. A. Jacobs, and B. Van Essen, "Communication Quantization for Data-Parallel Training of Deep Neural networks," in *Proceedings of the 2nd Workshop on Machine Learning in HPC Environments*, Salt Lake City, UT, USA, November 2016.

[15] C. Chen, J. Choi, D. Brand, A. Agrawal, W. Zhang, and K. Gopalakrishnan, "Adacomp: Adaptive Residual Gradient Compression for Data-Parallel Distributed Training," vol. 32, 2017.

[16] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," in *Proceedings of the INTERSPEECH 2015 16th Annual Conference Of the*

International Speech Communication Association, pp. 1488–1492, Dresden, Germany, 2015.

[17] A. Aji and K. Heafield, "parse Communication for Distributed Gradient Descent," in *S*, https://arxiv.org/abs/1704.05021, 2017.

[18] T. Chen, G. Giannakis, T. Sun, and W. Yin, "LAG: Lazily aggregated gradient for communication efficient distributed learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[19] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1310–1321, Denver, USA, October 2015.

[20] M. Arif, F. Ajesh, S. Shamsudheen, and M. Shahzad, "Secure and Energy-Efficient Computational Offloading Using LSTM in Mobile Edge Computing," *Security And Communication Networks*, vol. 2022, Article ID 4937588, 13 pages, 2022.

[21] M. Arif and F. Zaffar, "Challenges in efficient Data warehousing," *International Journal of Grid and Distributed Computing*, vol. 8, no. 2, pp. 37–48, 2015.

[22] Z. Jie, S. Chen, J. Lai, M. Arif, and Z. He, "Personalized Federated Recommendation System with Historical Parameter clustering," *Journal Of Ambient Intelligence And Humanized Computing*, pp. 1–11, 2022.

[23] X. Lu, Y. Liao, P. Lio, and P. Hui, "Privacy-preserving asynchronous federated learning mechanism for edge network computing," *IEEE Access*, vol. 8, pp. 48970–48981, 2020.