IEICE TRANS. INF. & SYST., VOL.E105–D, NO.2 FEBRUARY 2022

PAPER Special Section on Blockchain Systems and Applications

Layer-Based Communication-Efficient Federated Learning with Privacy Preservation*

Zhuotao LIAN[†], **Weizheng WANG**[††], **Huakun HUANG**[†††], ***Nonmembers***, ***and*** **Chunhua SU**[†a)], ***Member***

SUMMARY In recent years, federated learning has attracted more and more attention as it could collaboratively train a global model without gathering the users' raw data. It has brought many challenges. In this paper, we proposed layer-based federated learning system with privacy preservation. We successfully reduced the communication cost by selecting several layers of the model to upload for global averaging and enhanced the privacy protection by applying local differential privacy. We evaluated our system in non independently and identically distributed scenario on three datasets. Compared with existing works, our solution achieved better performance in both model accuracy and training time.

key words: *federated learning, privacy preservation, parameter selection, communication-efficient, non-IID data*

1. Introduction

In recent years, mobile devices have been widely used, and a large amount of data is generated locally. It is usually valuable but difficult for third parties to obtain as it can lead to privacy leakage. Thanks to the improvement in both computing power and storage capacity, training on mobile devices has become possible. In 2017, Google proposed federated learning, which allows clients to train on mobile devices with local data and protect personal privacy by only uploading the training result (e.g., gradients or parameters) and also designed Federated Averaging (FedAvg) algorithm for parameter updating [2].

There is usually one centralized parameter server and many clients (e.g., smartphones or IoT devices) in a federated learning system. Firstly, each client trains the local model, which is initialized at the beginning with local data and generates a new local model. Secondly, clients will send local models to the parameter server for averaging. The server aggregates all updates, generates a new global model and sends it back to all clients to update their local models. The process repeats until reaching the number of epochs or the set accuracy.

It brings many challenges as its distributed characteristics. In reality, the network condition for each client is different. Although network and communication technology have seen a rapid increase, it is still a significant concern in federated learning due to the heterogeneity among mobile devices.

Based on the problems mentioned above, we consider both the communication cost and privacy preservation in federated learning. We designed a novel layer-based communication-efficient federated learning system with privacy preservation which is illustrated in Fig. 1. To simplify the description, we call it our system in the following.

In our system, we proposed layer-based parameter selection method based on the basic structure of convolution neural network (CNN) to realize communication-efficient. Moreover, we enhanced privacy protection by adding artificial noise after the selection process, as Fig. 1 shows. We use local differential privacy mechanism in our simulation experiment. Once the layers are selected, clients will add noise to the selected part of model for privacy concerns. After that, these parameters will be sent to the server to update the global model.

There is an related study called Communication-Mitigated Federated Learning (CMFL) in [3]. Please note that they do parameter selection in a model sight, which means they will select a whole model for transmission or discard the whole model. They decide that whether the whole local model or, in other words, the client will take part in the aggregation step or not by computing the relevance between the local model and global model. The communication time could hardly be reduced because it always depends on the slowest client in each round of updates, and the participants still need to upload all model parameters. That is why we say this method is inefficient and not fine-grained enough. However, in our layer-based parameter selection method, we can shorten the communication time as the layer-based selection method could reduce the update data size by selecting several layers to transmit compared with CMFL.

In addition to the above two problems in federated learning, training data generated locally is always non-independent and identically distributed (non-IID), and it is mentioned in [4] that keep fitting models with non-IID data on different devices will eventually degrade the training performance. We design this system to be very flexible. Therefore, we can combine well with the existing research on non-

Manuscript received May 19, 2021.
Manuscript revised August 29, 2021.
Manuscript publicized September 28, 2021.

[†]The authors are with the Division of Computer Science, University of Aizu, Aizuwakamatsu-shi, 965–8580 Japan.
[††]The author is with the Department of Computer Science, City University of Hong Kong, Hong Kong.
[†††]The author is with School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, China.

*This paper is an extended version of work published in [1]. We extend our previous work to improve the papers' structure and have more evident results and discussions in non-IID (Independent and Identically Distributed) scenarios.

a) E-mail: chsu@u-aizu.ac.jp (Corresponding author)

DOI: 10.1587/transinf.2021BCP0006

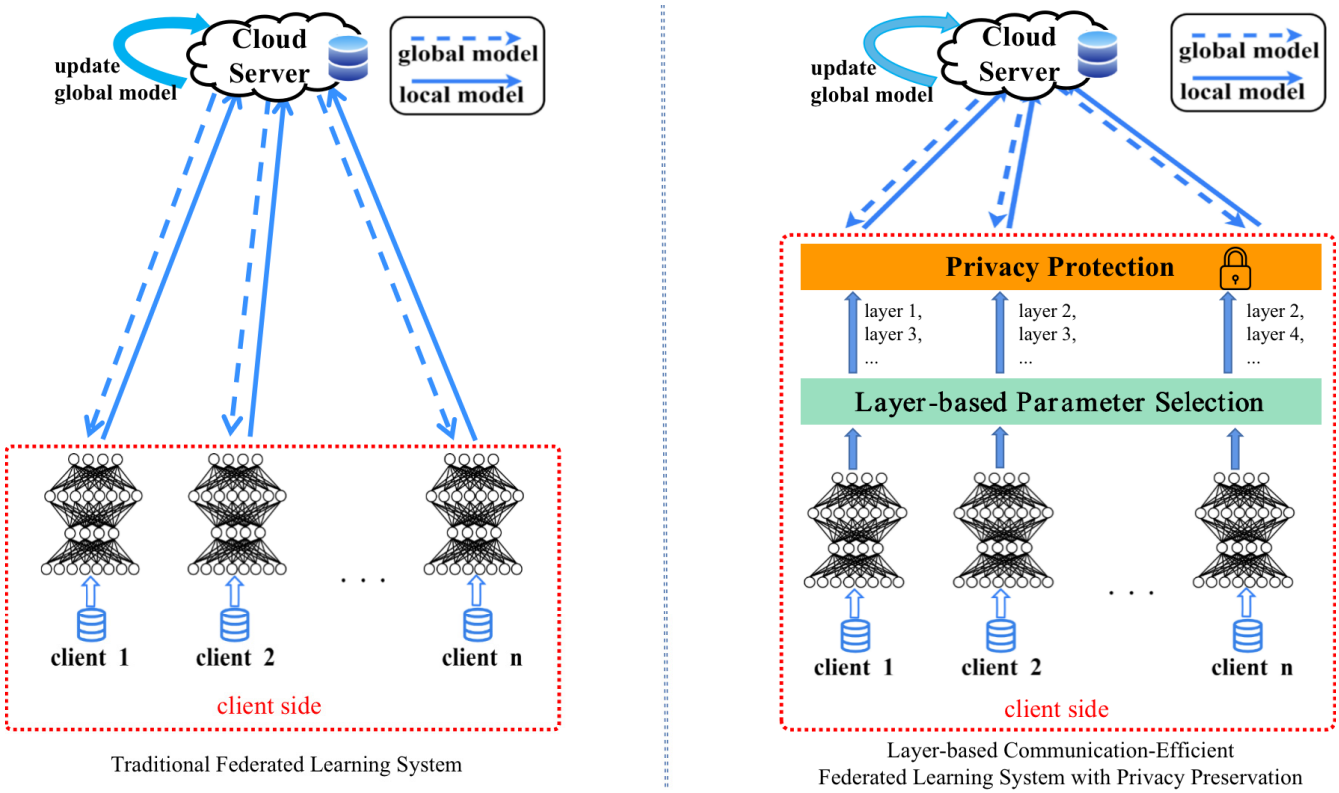


Fig. 1 Comparison of traditional federated learning system and our system

IID training.

We performed experiments on MNIST, Fashion-MNIST, and CIFAR-10. The non-IID datasets based on the above three were made to test the non-IID senior. Compared with traditional federated learning (FL) and CMFL, the results supported our work and showed the effectiveness of our system.

Our main contributions are summarized as follows:

1. A novel layer-based communication efficient federated learning system with privacy protection was proposed. Our system can shorten the communication time and reduce the amount of data transferred by the parameter selection process. Moreover, it can enhance users' privacy by applying local differential privacy.
2. We designed a layer-based parameter selection algorithm based on CMFL. It is more fine-grained than the one size fits all method in CMFL, and it could significantly reduce the update data size of each client, thus accelerating the whole training process. What is more, the uploaded parts of local models are different for each client due to the above process. We also designed layer-based AVG algorithm for global aggregation.
3. Performance evaluation was performed via simulation experiments of our system in both IID and non-IID scenarios. We tested on MNIST, Fashion-MNIST, and CIFAR-10 datasets and analyzed the results carefully.

We structured the reminders as follows: In Sect. 2, we discuss related researches. Section 3 presents the overall system design. Section 4 presents the parameter selection and privacy protection mechanisms and related algorithms. In Sect. 5, we conduct experiments on both IID and non-IID data to evaluate our system. Moreover, in Sect. 6, we give the conclusion.

2. Related Works

2.1 Federated Learning

Google Input Method uses federated learning to do the next word prediction [5] and also to do the emoji-prediction [6]. BrainTorrent was proposed to train convolutional neural networks on medical pictures without gathering the privacy-sensitive medical data [7]. Similarly, a federated learning framework was proposed for securely accessing and meta-analyzing any biomedical data by investigating brain structural relationships [8]. In financial fields, a federated learning-based method was proposed to collaboratively train a shared prediction model for credit risk management [9].

2.2 Communication Cost in Federated Learning

In [10], they explored Deep gradient compression technology. In [11], they proposed FedCS to select clients based on their resource conditions while in [3], they proposed Communication-Mitigated Federated Learning (CMFL) to do clients selection.

2.3 Privacy Preservation in Federated Learning

Although federated learning has avoided the sharing of clients' raw data, other uploaded data such as model weights or gradients may still lead to privacy leaks [12]. Practical Private Aggregation in Federated Learning with Local Differential Privacy (LDP-FL) was proposed in [13]. Similarly, LDP-Fed was present to solve the problem of applying local differential privacy in federated learning model which consists of high dimensional, continuous values with high precision [14].

2.4 Training on Non-IID Data

Training on non-IID data will raise challenges both in data modeling and convergence analyzing of the training procedures [15]. In [16], they discussed the convergence guarantee on non-IID data with the assumption of uniformly bounded gradients. In addition, a data-sharing strategy was proposed to create a small subset of globally shared data to improve training performance with heterogeneous data [17].

3. System Design

We describe the models and design of our system in this section. To begin with, we will introduce the threat model in our system. In addition, specific modules of the server and the client will be discussed, functions and the workflow between them will also be declared. Specific to the algorithms of averaging, parameter selection, and privacy preservation, will be introduced in detail in Sect. 4.

3.1 Threat Model

The membership inference attack in federated learning could be mounted by an adversary who attempts to infer whether a specific piece of data is part of the training set of either a specific or any client [18]. In reality, the central server could also be an adversary to reveal private information through parameter updates, and we consider that both server and clients are untrustworthy.

3.2 Modules in the System

There is a parameter server located in the cloud in our system, and there are n clients participating in training. The client means mobile devices, which could be smartphones, computers, IoT devices, and so on. As Fig. 2 illustrates, the system could be divided into two sides, the client side and the server side.

On the client side, each client mainly has three modules, namely trainer, selector, and protector. Trainer is the first module and the head of the workflow on mobile devices. It is responsible for training local models with local data and maintaining the models. After that, the local model

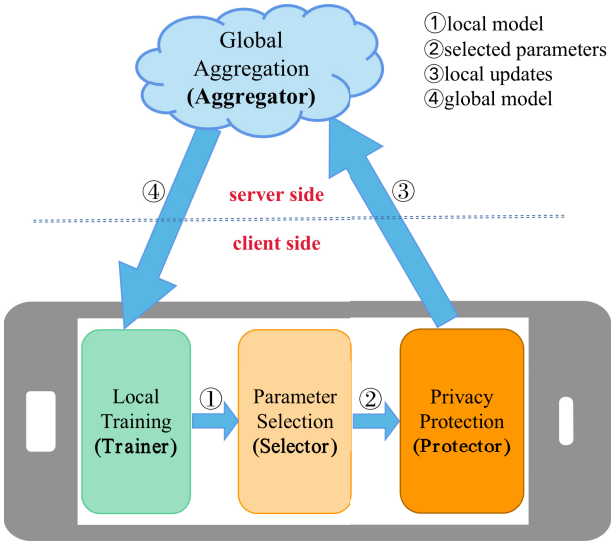


Fig. 2 System design

will be sent to selector. Selector is responsible for layer-based parameter selection. Then, the selected layers of the model will be passed to protector. Protector will perturb the data and then send local updates (selected and processed part of local model) to the parameter server.

On the server side, there is one key module called aggregator. The aggregator will weighted average the local updates received from clients to generate a new global model and then send it back to all clients to update their local models.

4. Algorithms Design

Here we will firstly introduce the Layer-based AVG algorithm. And then we will focus on the parameter selection and privacy preservation algorithms. The principally used symbols were listed with the notations in Table 1.

4.1 Layer-Based AVG Algorithm

Due to the distributed nature and low bandwidth among mobile clients, the amount of data transmitted and the communication delay could cause huge communication overhead, which could be a bottleneck in federated training [19]. In [2], FedAVG was proposed. It allows clients to communicate with the server after training for several rounds locally. Based on this, we designed Layer-based AVG algorithm.

In the beginning, we randomly select m clients to participant in this epoch. Each client will train their local models with their own data by gradient descent. $\boldsymbol{w}_i(t)$ means the whole local model of client i at epoch t.

As Algorithm 2 illustrates, each client will train their local models for k rounds. After that, the training result $\boldsymbol{w}_i(t)$ will be regarded as the input of parameter selection.

Later, each client will perform layer-based parameter selection to select the parameters for global averaging and generate $\boldsymbol{w}'_i(t)$. Moreover, they will do data perturbation to

Table 1 Variables and symbols

$\mathtt{m}$	The number of participants in each epoch.
k	Each client trains their local models for k rounds in each epoch.
$\boldsymbol{w}_i(t)$	Local model of client i at epoch t.
$\boldsymbol{w}'_i(t)$	The selected part of model of client i at epoch t.
$\widetilde{\boldsymbol{w}}'_i(t)$	The model after data perturbation on $\boldsymbol{w}'_i(t)$.
$\widetilde{\boldsymbol{w}}'_{ij}(t)$	Layer j of the perturbed model $\widetilde{\boldsymbol{w}}'_i(t)$.
$\boldsymbol{w}_j(t)$	Layer j of the global model in epoch t.
$\lvert D_i\rvert$	The total number of training samples of client i.
$L_j(t)$	The set consists of the clients who have selected layer j for global aggregation at epoch t.

Algorithm 1 Layer-based AVG

1: Initialize all clients' local models with $\boldsymbol{w}_0$
2: **for** epoch $t = 1, 2, \ldots, T$ **do**
3: Randomly select m clients
4: **for** each client $i \in [m]$ in parallel **do**
5: Local Update($\boldsymbol{w}_i(t-1)$)
6: $\boldsymbol{w}'_i(t) \leftarrow$ Layer-based Parameter Selection($\boldsymbol{w}_i(t)$)
7: $\widetilde{\boldsymbol{w}}'_i(t) \leftarrow$ Data Perturbation($\boldsymbol{w}'_i(t)$)
8: **end for**
9: **for** each layer j of model in parallel **do**
10: $\boldsymbol{w}_j(t) \leftarrow \frac{\sum_{i \in L_j(t)} \lvert D_i\rvert \widetilde{\boldsymbol{w}}'_{ij}(t)}{\sum_{i \in L_j(t)} \lvert D_i\rvert}$
11: **end for**
12: $\boldsymbol{w}(t) \leftarrow$ combine each layer's parameters $\boldsymbol{w}_j(t)$
13: **for** each client $i \in [n]$ in parallel **do**
14: $\boldsymbol{w}_i(t) \leftarrow \boldsymbol{w}(t)$
15: **end for**
16: **end for**

Algorithm 2 Local Update

1: **Input:** $\boldsymbol{w}_i(t-1)$, k
2: Initialize variable c with hyperparameter k
3: $c = k$
4: **while** $c > 0$ **do**
5: $\boldsymbol{w}_i(t-1) \leftarrow \boldsymbol{w}_i(t-1) - \eta \nabla F_i(\boldsymbol{w}_i(t-1))$
6: $c = c - 1$
7: **end while**
8: **return** $\boldsymbol{w}_i(t-1)$

protect privacy and generate $\widetilde{\boldsymbol{w}}'_i(t)$ to transfer.

After gathering all clients' updates, a new global model will be generated at the server. As line 9 of Algorithm 1 denotes, the weighted averaging is also performed in a layer sight instead of the whole model. It can be easily extended from the traditional parameter averaging method in federated learning. Finally, the parameter server will send it back to all clients to update their local models for the next epoch of training.

4.2 Layer-Based Parameter Selection Algorithm

This section will explain the design of layer-based parameter selection and the difference between our method and that in Communication-Mitigated Federated Learning (CMFL).

Convolution neural network (CNN) could be divided into mainly convolutional layers (CONV), pooling layers (POOL), activation layers (ReLU) and fully connected layers (FC), etc [20]–[22]. We designed our layer-based parameter selection method based on this structure.

Algorithm 3 Layer-based Parameter Selection

1: **Input:** $\boldsymbol{w}_i(t)$, $A(t)$
2: **for** each layer j in model $\boldsymbol{w}_i(t)$ in parallel **do**
3: Calculate $r(\boldsymbol{w}_{ij}(t), \boldsymbol{w}_j(t))$ by following Eq. (3)
4: **if** $\mathrm{r}(\boldsymbol{w}_{ij}(t), \boldsymbol{w}_j(t)) > A(t)$ **then**
5: Layer j is selected for global aggregation
6: Add i into $L_j(t)$
7: **end if**
8: **end for**
9: $\boldsymbol{w}'_i(t) \leftarrow$ Selected layers
10: **return** $\boldsymbol{w}'_i(t)$

We designed layer-based parameter selection based on the method in Communication-Mitigated Federated Learning (CMFL) [3]. In CMFL, they performed selection by checking the relevance of the local and global model [3]. Next, we will explain the difference between our method and the existing one.

In CMFL system, they simply compared the whole local model with the global model by Eq. (1) and (2).

$$r(\boldsymbol{w}, \bar{\boldsymbol{w}}) = \frac{1}{N} \sum_{k}^{N} I(sgn(e_k), sgn(\bar{e}_k)) \tag{1}$$

$$I(sgn(e_k), sgn(\bar{e}_k)) = \begin{cases} 1, & \text{if } e_k \text{ and } \bar{e}_k \text{ are of} \\ & \text{the same sign} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$\boldsymbol{w}$ and $\bar{\boldsymbol{w}}$ denote the global model and local model while e_k means the element k of the model. $A(t)$ is the fine-tuned threshold. There are N elements in total. From the equations, we can see that it is a one size fit all method in CMFL, and it has to be decided whether to transmit the whole model. However, different layers of neural network models have different characters and differ in relevance and training performance. In order to verify our idea, we did verification experiments on MNIST with Eq. (1) which showed that for the parameters of different layers, the relevance between them and the corresponding layer of the global model is also different. So it is necessary to evaluate the relevance in a layer sight.

Compared with that, our layer-based method uses Eq. (3) to check the relevance.

$$r(\boldsymbol{w}_j, \bar{\boldsymbol{w}}_j) = \frac{1}{N_j} \sum_{k}^{N_j} I(sgn(e_k), sgn(\bar{e}_k)) \tag{3}$$

Similarly, $\boldsymbol{w}_j$ and $\bar{\boldsymbol{w}}_j$ mean the layer j of global model and local model while e_k means the element k of this layer. N_j denotes the number of elements in layer y. And our layer-based parameter selection has the following advantages.

1. We can check the relevance between local model and global model in a more fine-grained way as Eq. (3) illustrates. Besides, the simulation results in Sect. 5 also

Algorithm 4 Data Perturbation

1: **Input:** $\boldsymbol{w}_i'(t)$, function $l(x)$
2: **for** each element $e \in \boldsymbol{w}_i'(t)$ **do**
3: $\quad \widetilde{e} \leftarrow l(e)$
4: $\quad$ Replace e as $\widetilde{e}$
5: **end for**
6: Then, we have
7: $\widetilde{\boldsymbol{w}}_i'(t) \leftarrow \boldsymbol{w}_i'(t)$
8: **return** $\widetilde{\boldsymbol{w}}_i'(t)$

pointed that our layer-based method could reach better performance.
2. Our layer-based method can significantly reduce the communication time by selecting several layers to upload instead of the whole local model.
3. Our method can improve the stability of the federated learning system. Selecting part of the local model to transmit and perturb could reduce computing overhead and thus can effectively reduce power consumption, which is very meaningful for mobile devices.

4.3 Data Perturbation Algorithm

Generally, differential privacy and local differential privacy (LDP) differ in definition and whether the noise is added locally on the client side or the server side [13]. LDP has been applied in privacy-preserving data collection by many companies to deal with the inference attacks against shared model parameters [14].

Definition 1: (ϵ-*LDP*). Where $\epsilon > 0$, a randomized algorithm l satisfies ϵ-local differential privacy, if and only if for any inputs e_1, e_2 in the finite universe of possible values for user data e, for any possible output s, we have:

$$\frac{Pr[l(e_1) = s]}{Pr[l(e_2) = s]} \leq e^{\epsilon} \tag{4}$$

In LDP settings, users upload their perturbed data which is guaranteed to protect x from inference attacks instead of the original data. According to the definition, a lower ϵ can guarantee a higher level of privacy.

As Algorithm 4 illustrates, we do data perturbation on the selected parameters on the client side by applying Laplace mechanism, which is a general method for preserving ϵ-*LDP* [23]. In line 3 of Algorithm 4, e is the value of the element in $\boldsymbol{w}_i'(t)$. Function $l(x)$ is defined as follows:

$$l(x) = x + Lap(\Delta s/\epsilon) \tag{5}$$

where $\frac{\Delta s}{\epsilon}$ is the scale parameter, Δs is the local sensitivity of clients, and $Lap(\cdot)$ is the Laplace distribution.

5. Experiments

We evaluated our system on three non-IID datasets, namely MNIST, Fashion-MNIST, and CIFAR-10. We detailed the settings of our experiments, then evaluated the performance according to the results.

5.1 Settings

We compared our system with Communication-Mitigated Federated Learning (CMFL) and traditional federated learning, which is considered the baseline. To simplify the description, we call the above our system, CMFL, and FL.

We assume one parameter server and 100 clients participating in the Conventional Neural Network (CNN) training process. In order to control the variables to compare the performance of the system, we specify that the number of randomly selected clients during the training process is 70, and it will not change. Moreover, the default value of k in Algorithm 2 is one unless we specify it. Also, we applied LDP to both conventional federated learning system and CMFL. The data perturbation setting is the same as that in our system. For simplicity, LDP is included by default when we mention CMFL or FL. We use $\epsilon = 10$ when we compare the performance of the above three systems for privacy concerns. According to our test in previous work [1], the thresholds we chose for experiments on MNIST, Fashion-MNIST, and CIFAR-10 are 0.75, 0.65, and 0.4. Then, we use these thresholds to perform the following experiments.

The datasets and models we used are as follows:

1. MNIST dataset is comprised of 10-class handwritten digits. We consider a CNN model with two convolution layers. The shape of the first one is 26×26, and it has 32 output channels. The second one is 24×24 with 64 output channels. After that, there is a 12×12 max pooling layer with 64 output channels.
2. Fashion-MNIST has the same data size and format as MNIST [24]. Here we train the same CNN model on Fashion-MNIST.
3. CIFAR-10 has 50000 training images, and 10000 testing images [25]. We train a CNN model with four convolution layers. The first one is 32×32 while the second one is 30×30 with the same 32 output channels. Then, there is a 15×15 max pooling layer with 32 output channels. The third convolution layer is 15×15 while the fourth is 13×13 with the same 64 output channels. After that, there is a 6×6 max pooling layer with 64 output channels.

Our system was evaluated on different non-IID levels of three datasets. For the non-IID settings, we introduce the variable a. When $a = 1.0$, it denotes that each client has only one type of data. And $a = 0.7$ means 70% data of each client belong to one class, and the remaining data belong to other classes. When $a = 0$, the data is independent and identically distributed.

Finally, we explored the existing solutions to non-IID problem in federated learning, and conducted experiments through the pre-training model method proposed in [17].

5.2 Results on Non-IID Training

The three datasets all consist of 10 classes of data. The

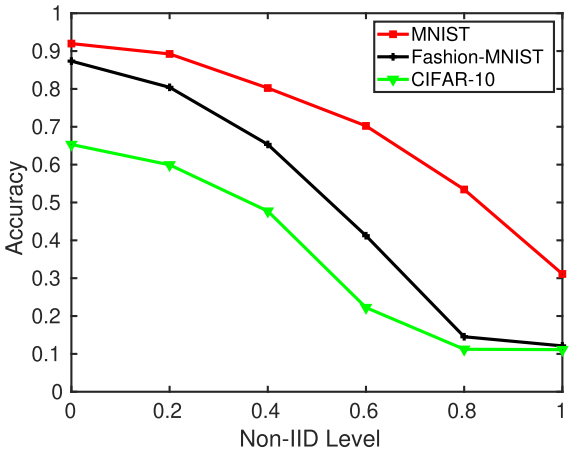


Fig. 3 Accuracy versus non-IID level.

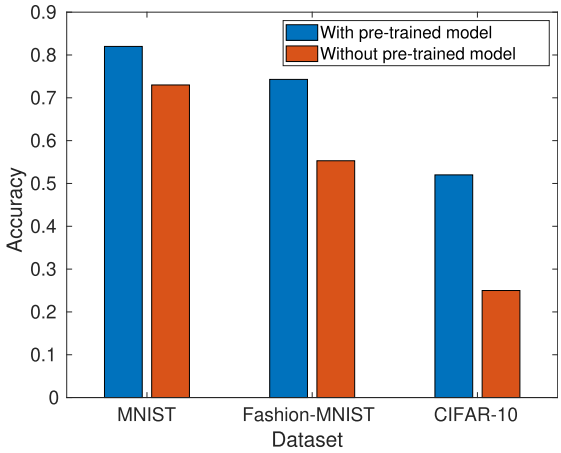


Fig. 4 Performance comparison for whether the model is pre-trained on IID data.

Table 2 Accuracy versus non-IID level a among different federated learning systems.

	a=0	a=0.3	a=0.7	a=1
Our system	0.8502	0.7630	0.3527	0.1100
CMFL	0.8356	0.7206	0.3223	0.1002
FL	0.8213	0.7497	0.4232	0.1101

model we trained is to infer which category the input image belongs to. And accuracy refers to the correct rate of inference on the test set.

Figure 3 compares the impact of different non-IID levels on three datasets. We tested on conventional FL and noted the accuracy of the model after training for 50 epochs. No matter which dataset, the accuracy of our system will decrease as the non-IID level increases and the downward trend of accuracy is getting faster and faster.

$a = 0$ denotes that the data of each client is independent and identically distributed. In our datasets settings, it means each client has ten classes of data. The accuracy of the model is relatively high compared with other non-IID levels. When $a = 1$, it is almost unable to train. The accuracy is approximately equal to 0.1, which is close to randomly selecting.

In Table 2, we tested our system, CMFL and FL on Fashion-MNIST at the same time. Similarly, the result after training for 2000 seconds shows that they all have the same trend.

The performance of the above three systems drops sharply with the increase of a, and they even reach the point where it is unable to train when $a = 1$. We can see that non-IID is quite a challenge in federated learning and could do great harm to existing systems. Despite this, our system still got better performance in the seniors that $a = 0$ and $a = 0.3$. It reached an accuracy of 0.8502 and 0.763, while CMFL got 0.8356 and0.7206. That is because our system has a more fine-grained layer-based parameter selection method, and thus saved the communication time and reach a higher accuracy after training for the same time.

With the growth of non-IID levels, the performance gap between these systems will gradually narrow and is even inferior to traditional federated learning. When completely independent and identically distributed, the accuracy of these systems is almost the same. The data distribution is too uneven, systems' performance is greatly affected, and the models' training may proceed in an unexpected direction so that it cannot fit well in the continuous training process.

We can clearly see that non-IID is quite a challenge in federated learning and could do great harm to existing systems. We will next try the pre-trained model method in our system.

5.3 Non-IID Training with Pre-Trained Model

In [17], they proposed a strategy for non-IID training by creating a globally shared small subset of data to generate a pre-train model. Here we applied this method to our system.

To start with, we need to gather a globally shared dataset. Here we let each client share 10% data to generate an IID subset. In real life, there are various ways to construct an auxiliary dataset. For example, interconnected smart devices in the house can share some data while protecting privacy. Another example is that clients who trust each other can share some data. Moreover, a large amount of data is swapped between heterogeneous sensors and devices every moment as mentioned in [26].

It is also possible to make a dataset more reasonably based on each person's data distribution. Here we will not go deep but focusing on the performance of the pre-trained method in our system.

We tested on three datasets with the non-IID level $a = 0.6$. To conduct the pre-trained process, we kept fitting the model on the auxiliary dataset for ten epochs. After that, we used it to initialize the local models of all clients. And then, the federated learning process started and repeated for 50 epochs.

As Fig. 4 shows, training with a pre-trained model

could always reach a better performance on non-IID data. When the dataset is more challenging to train, the improvement of the pre-training model method is more significant. The accuracy on CIFAR-10 is about two times higher than that without the pre-trained process, while it improved around 14% and 36% accuracy on MNIST and Fashion-MNIST, respectively.

6. Conclusion

Communication cost and non-IID data have brought significant challenges in federated learning. In this paper, we presented layer-based communication-efficient federated learning with privacy preservation. We designed a layer-based parameter selection method and the layer-based AVG algorithm, which have significantly reduced the communication cost according to the experiment result. What is more, we also applied local differential privacy to enhance privacy protection. Finally, we performed experiments in the non-IID scenario and tested with a pre-trained model method to show the effectiveness of our system.

In future work, we will explore more reasonable parameter selection methods, mainly the setting of thresholds. We will explore more adaptable threshold determination methods that can change autonomously with the training process and effects. Moreover, differential privacy is essentially a trade-off between security and training effects. How to make better trade-offs to apply to a broader range of scenarios is also the content of our follow-up research.

Acknowledgments

This work was partly supported by the National Natural Science Foundation of China under Grant 62072485 and Grant 62001126.

References

[1] Z. Lian, W. Wang, and C. Su, "COFEL: Communication-Efficient and optimized federated learning with local differential privacy," 2021 IEEE International Conference on Communications (ICC): Communication and Information Systems Security Symposium (IEEE ICC'21 - CISS Symposium), Montreal, Canada, June 2021.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B.A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," Artificial Intelligence and Statistics, pp.1273–1282, 2017.

[3] L. Wang, W. Wang, and B. Li, "Cmfl: Mitigating communication overhead for federated learning," 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), pp.954–964, IEEE, 2019.

[4] M. Mohri, G. Sivek, and A.T. Suresh, "Agnostic federated learning," arXiv preprint arXiv:1902.00146, 2019.

[5] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," 2018.

[6] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," arXiv preprint arXiv:1906.04329, 2019.

[7] A.G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," 2019.

[8] S. Silva, B.A. Gutman, E. Romero, P.M. Thompson, A. Altmann, and M. Lorenzi, "Federated learning in distributed medical databases: Meta-analysis of large-scale subcortical brain data," 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019), pp.270–274, 2019.

[9] D. Kawa, S. Punyani, P. Nayak, A. Karkera, and V. Jyotinagar, "Credit risk assessment from combined bank records using federated learning," 2019.

[10] Y. Lin, S. Han, H. Mao, Y. Wang, and W.J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," arXiv preprint arXiv:1712.01887, 2017.

[11] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," ICC 2019-2019 IEEE International Conference on Communications (ICC), pp.1–7, IEEE, 2019.

[12] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," 2019 IEEE Symposium on Security and Privacy (SP), pp.691–706, IEEE, 2019.

[13] L. Sun, J. Qian, X. Chen, and P.S. Yu, "Ldp-fl: Practical private aggregation in federated learning with local differential privacy," arXiv preprint arXiv:2007.15789, 2020.

[14] S. Truex, L. Liu, K.H. Chow, M.E. Gursoy, and W. Wei, "Ldp-fed: federated learning with local differential privacy," Proc. Third ACM International Workshop on Edge Systems, Analytics and Networking, pp.61–66, 2020.

[15] T. Li, A.K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," arXiv preprint arXiv:1908.07873, 2019.

[16] H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," Proc. AAAI Conference on Artificial Intelligence, vol.33, pp.5693–5700, 2019.

[17] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," arXiv preprint arXiv:1806.00582, 2018.

[18] E. De Cristofaro, "An overview of privacy in machine learning," arXiv preprint arXiv:2005.08679, 2020.

[19] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," Advances in Neural Information Processing Systems, pp.9850–9861, 2018.

[20] S. Wang, D. Li, and J. Geng, "Geryon: Accelerating distributed cnn training by network-level flow scheduling," IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pp.1678–1687, IEEE, 2020.

[21] D. Vasan, M. Alazab, S. Wassan, B. Safaei, and Q. Zheng, "Image-based malware classification using ensemble of cnn architectures (imcec)," Computers & Security, vol.92, p.101748, 2020.

[22] T.R. Gadekallu, M. Alazab, R. Kaluri, P.K.R. Maddikunta, S. Bhattacharya, K. Lakshmanna, and M. Parimala, "Hand gesture classification using a novel cnn-crow search algorithm," Complex & Intelligent Systems, vol.7, no.4, pp.1855–1868, 2021.

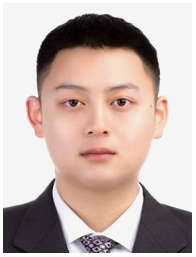
[23] N. Phan, X. Wu, H. Hu, and D. Dou, "Adaptive laplace mechanism: Differential privacy preservation in deep learning," 2017 IEEE International Conference on Data Mining (ICDM), pp.385–394, IEEE, 2017.

[24] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," arXiv preprint arXiv:1708.07747, 2017.

[25] A. Krizhevsky, G. Hinton, et al., "Learning multiple layers of features from tiny images," 2009.

[26] W. Wang, H. Xu, M. Alazab, T.R. Gadekallu, Z. Han, and C. Su, "Blockchain-based reliable and efficient certificateless signature for iiot devices," IEEE Trans. Ind. Informat., 2021.

Zhuotao Lian received his B.S. degree in Computer Science and Technology from China University of Geosciences, Wuhan, China, in 2020. He is currently a second year graduate school student in the School of Computer Science and Engineering, the University of Aizu, Fukushima, Japan. His research interests mainly focus on distributed system, machine learning, information security and edge computing.



Weizheng Wang received the B.S. degree in software engineering from Yangzhou University, Yangzhou, China, in 2019, the M.S. degrees in computer science and engineering from the University of Aizu, Aizu-Wakamatsu, Japan, in 2021. Now he is currently a Research Associate in University of Aizu and pursuing the Ph.D. degree in computer science at the City University of Hong Kong, Hong Kong. His research interests include applied cryptography, blockchain technology and IoT system.

Huakun Huang received the Ph.D. in Computer Science and Engineering from the University of Aizu, Japan, in 2019. He is currently an associate professor with the School of Computer Science and Cyber Engineering, Guangzhou University, China. His current research interests include privacy preserving, machine learning, federated learning and continuous intelligent networks.

Chunhua Su received the B.S. degree for Beijing Electronic and Science Institute in 2003 and received his M.S. and Ph.D. of computer science from Faculty of Engineering, Kyushu University in 2006 and 2009, respectively. He is currently working as a Senior Associate Professor in Division of Computer Science, University of Aizu. He has worked as a postdoctoral fellow in Singapore Management University from 2009-2011 and a research scientist in Cryptography & Security Department of the Institute for Infocomm Research, Singapore from 2011-2013. From 2013-2016, he has worked as an Assistant professor in School of Information Science, Japan Advanced Institute of Science and Technology. From 2016-2017, he worked as Assistant Professor in Graduate School of Engineering, Osaka University. His research interests include cryptanalysis, cryptographic protocols, privacy-preserving technologies in machine learning and IoT security & privacy. He has published more than 100 papers in international journals and conferences.