MSRA-Fed: A Communication-Efficient Federated Learning Method Based on Model Split and Representation Aggregate

LIU Qinbo[1,2], JIN Zhihao[1], WANG Jiabo[1], LIU Yang[1,3], LUO Wenjian[1,3]

(1. School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China;
2. Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies, Shenzhen 518055, China;
3. Peng Cheng Laboratory, Shenzhen 518055, China)

DOI: 10.12142/ZTECOM.202203005

https://kns.cnki.net/kcms/detail/34.1294.TN.20220804.1854.002.html, published online August 5, 2022

Manuscript received: 2022-06-20

Abstract: Recent years have witnessed a spurt of progress in federated learning, which can coordinate multi-participation model training while protecting the data privacy of participants. However, low communication efficiency is a bottleneck when deploying federated learning to edge computing and IoT devices due to the need to transmit a huge number of parameters during co-training. In this paper, we verify that the outputs of the last hidden layer can record the characteristics of training data. Accordingly, we propose a communication-efficient strategy based on model split and representation aggregate. Specifically, we make the client upload the outputs of the last hidden layer instead of all model parameters when participating in the aggregation, and the server distributes gradients according to the global information to revise local models. Empirical evidence from experiments verifies that our method can complete training by uploading less than one-tenth of model parameters, while preserving the usability of the model.

Keywords: federated learning; communication load; efficient communication; privacy protection

Citation (IEEE Format): Q. B. Liu, Z. H. Jin, J. B. Wang, et al., "MSRA-Fed: a communication-efficient federated learning method based on model split and representation aggregate," *ZTE Communications*, vol. 20, no. 3, pp. 35–42, Sept. 2022. doi: 10.12142/ZTECOM.202203005.

1 Introduction

As the Internet has found its way into our everyday life, the deep integration of the artificial intelligence technology represented by deep learning and IoT technology has become a new trend, and then the concept of Artificial Intelligence of Things (AIoT) [1] came into being. AIoT not only requires each device to be intelligent, but also that intelligent terminals can cooperate and integrate with each other, so as to give full play to the great value of IoT. AIoT already has a wide range of applications. For example, the intelligent camera can do object tracking and detect abnormal sound according to real-time voice; the smart bracelet can analyze users' health status according to the monitored data; intelligent systems in autonomous vehicles collect data from monitoring equipment to make driving decisions. However, traditional AI techniques usually require centralized data collection and centralized training of models, which needs to upload the data on IoT devices to application providers. Such paradigms obviously raise privacy concerns. For example, there is a high probability that users will not allow smart security cameras to upload their bedroom surveillance video to the server for model training[2]. Therefore, the practical deployment of AIoT faces the challenge of privacy risks.

To alleviate privacy concerns, an effective approach is to employ federated learning[3]. Federated learning stipulates that each client saves data locally, and only uses parameters instead of raw data to communicate between clients and the server. Therefore, it can be used as a privacy-preserving computing paradigm and has received extensive attention in academia and industry. Especially in the field of IoT, federated learning has been proven to power IoT in data sharing, attack detection, mobile group perception, privacy and security[4]. With federated learning, each IoT device is no longer limited to acquiring knowledge from its own dataset, but can benefit

This work is supported by Shenzhen Basic Research (General Project) under Grant No. JCYJ20190806142601687, Shenzhen Stable Supporting Program (General Project) under Grant No. GXWD20201230155427003-20200821160539001, Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies under Grant No. 2022B1212010005, and Shenzhen Basic Research (Key Project) under Grant No. JCYJ20200109113405927.
The corresponding author is LIU Yang. LIU Qinbo, JIN Zhihao and WANG Jiabo contribute equally in this work.

from the data and information of other devices while protecting their own privacy. Taking the widely used FedAvg algorithm[5] as an example, in each round of training, IoT clients randomly selected by the central server upload model parameters to the server. The central server only receives and aggregates these parameters to obtain an enhanced global model for distributed learning. Then, the central server distributes the aggregated model to guide local training. The data of each client are always stored locally, and knowledge sharing among IoT devices is achieved through the aggregation of model parameters.

However, there are still many challenges for federated learning to be deployed on IoT devices, one of which is the urgent need to improve communication efficiency. Although the iterative development of communication technologies such as 4G and 5G in recent years has made the bandwidth quite impressive, compared with the computing performance of the server and clients, what hinders training efficiency of federated learning the most is the communication between the server and clients[6]. Especially in edge computing and IoT, there are some applications that require extremely high communication efficiency. In some practical settings where federated learning is deployed, in order to improve the overall operational efficiency, the system will ignore some devices with limited network bandwidth or limited access, that is, they will not participate in the training round. However, such simple processing will lead to some local models not being optimized, which will seriously affect user experience[7].

In order to improve the communication efficiency of federated learning, an effective way is to reduce the amount of communication data between clients and the server. FedProto[8] proposed to replace the gradient-based aggregation with prototype-based aggregation. The prototype size is much smaller than the size of gradient, so the prototype-based method can effectively improve communication efficiency. However, FedProto simply uploads the prototype to the server for weighted average to obtain global prototype, which cannot effectively fuse the information of clients. SplitFed[9] uses the idea of split learning to split the neural network into a client-side network and a server-side network, thereby reducing the parameters required for communication. However, multi-client split learning is done asynchronously, so it is inefficient and causes clients to be idle. Subsequently, SplitFed introduces a fed server to execute FedAvg on the client side, which can synchronously train the split learning and greatly accelerate convergence. However, this method requires clients to upload smash data and the server to send gradient back to the clients in each round of model update. The amount of data transferred between the server and clients is still very large.

To cope with the problem of high communication load in federated learning, we propose a novel method for efficient communication based on model split and representation aggregate—MSRA-Fed. MSRA-Fed considers the advantages of federated learning and split learning, and significantly reduces the amount of data communicated between clients and the server. It also provides stronger privacy protection than the traditional gradient-based communication. At the same time, the proposed method can ensure the deep fusion of information of each client during the aggregation process. The paradigm in this paper is suitable for federated learning scenarios that do not require particularly high accuracy, but require low communication costs and strict security.

Our contributions are as follows:

1) Through empirical evidence from experiments, we verify that the outputs of the last hidden layer of a neural network can carry the characteristics of the training set. Therefore, these outputs can be used to replace the parameters of the model to cooperate with each client for model training.

2) In response to the above observations, we propose a communication optimization strategy based on model split and representation aggregate. This approach can significantly reduce the parameters the client needs to upload, reduce the communication load of federated learning, and ensure the accuracy of the model.

The rest of this paper is organized as follows. Section 2 presents the background and related work. Section 3 introduces the scheme of MSRA-Fed in detail. Section 4 verifies the effectiveness of MSRA-Fed through experiments and shows the experimental results. Finally, a summary and discussion of future work are presented in Section 5.

2 Background and Related Work

Federated learning is a machine learning framework, whose purpose is to use distributed data to collectively train a common model[10]. In this way, the storage and computing power of each participant can be fully utilized. During the training process, decentralized clients will only have parameter communication with the central server, and no raw data will be exchanged between any clients[11]. Compared with the way where each participant trains independently, the participants in federated learning can obtain other client-side knowledge from the global model issued by the server, so as to make local models more effective. These characteristics of federated learning not only allow us to combine multi-party data for mining and analysis, but also avoid direct interaction of raw data and protect data privacy. Since federated learning was proposed, a large number of related papers and achievements have emerged. We note that there is extensive research on how to improve efficiency and effectiveness of federated learning.

MCMAHAN et al.[5] proposed FedAvg based on a centralized training architecture, which is robust to non-IID (independent and identically distributed) data distributions and can reduce the communication rounds required to train the model.

FedAvg adopts a synchronous update scheme during each round of training. For a fixed number of K clients, the server will randomly select a portion of the clients to participate in each round of training according to a fraction C. Ref. [5] shows the reward brought by increasing the number of clients will gradually decrease if the number of participating clients exceeds a certain threshold. In each round of local training, the client k calculates the gradient g_k of the local data under the current model to update its local model w_k. Typically, the client k can update w_k through multiple local epochs of training, and then the central server aggregates. Compared with the general distributed stochastic gradient descent, FedAvg reduces the number of iterations of global training by increasing the amount of computation on the client side, thereby reducing the communication rounds.

In the training process of FedAvg, each participant needs to frequently communicate parameters with the central server to update the local model. The amount of data communicated is generally large, resulting in a high communication cost[4]. WANG et al.[12] designed a Communication-Mitigated Federated Learning (CMFL) framework by identifying irrelevant updates on clients and excluding them in advance to prevent invalid updates, which aims to reduce communication costs by avoiding uploading irrelevant parameters. Although this method improves communication efficiency, it increases a lot of computational cost. SATTLER et al.[13] proposed an Sparse Ternary Compression (STC) compression framework for the shortcomings of several compression-based solutions that only compress upstream communication and are only effective for ideal conditions (such as IID data distribution). Experiments show that STC outperforms the FedAvg algorithm under certain conditions. Refs. [14] and [15] proposed two ways to reduce the cost of upstream communication: the structured update and summary update. The former uses fewer samples and fewer updates; the latter uses lossy compression to update parameters. However, both approaches lack robustness when dealing with poor quality data. CALDAS et al. [16] proposed a federated mechanism that makes it possible to efficiently train a smaller subset of the global model and address the downlink communication pressure with server-to-client compression. This work broke the situation that downlink communication has not been studied. The above-mentioned research reduces the amount of transmitted data by compressing the original model or obtaining more compact updates, although the accuracy decreases to some extent.

In order to communicate efficiently, there are also some studies that reduce the number of communication parameters by splitting the model into different devices. Our idea is primarily inspired by the scheme of model split. DEAN et al.[17] proposed DistBelief, which uses the paradigm of model parallelism by model split. DistBelief can manage the data transmission between participants in the process of bottom communication, synchronization, training and inference. However, since many clients will be idle when the system is running, DistBelief is not efficient. GPipe[18] proposed by Google introduces pipelines on the basis of model parallelism, which improves the utilization of devices in the parallelism. However, as a special setting of distributed learning, federated learning assumes that the central server is not allowed to manage and schedule the clients participating in the local training. Therefore, the conventional model split and pipeline parallelization are not suitable for federated learning. Recently, some researchers have also tried to introduce model split into federated learning. Ref. [19] applied split learning to Long Short-Term Memory (LSTM) networks and proposed LSTMSPLIT to classify time-series data with multiple clients. Ref. [20] proposed an asynchronous learning strategy, which divided the neural network into deep and shallow layers. The method of updating the shallow layers more frequently than the deep layers can reduce the communication cost. THAPA et al.[9] proposed SplitFed by combining federated learning and split learning[21], which solves the problem that each client cannot be updated synchronously under the model split strategy. However, the amount of data to be transmitted in these methods is still large and the communication frequency is too high for efficient training.

3 MSRA-Fed Method

3.1 Outputs of Last Hidden Layer Carry Characteristics

The trained model can reflect the characteristics of a training set and even remember training samples when overfitting[22]. That is, the model parameters carry the characteristics of the training samples. Therefore, the method adopted by federated learning when updating a global model is to upload the parameters of local models to the server for aggregation, in order to obtain a better global model without violating the privacy of training sets. However, the parameters of a neural network model are usually of high dimension, which makes clients suffer from high communication load in neural network based federated learning tasks. Therefore, we choose to use other data carrying the characteristics of training samples instead of all model parameters for server aggregation, to trade off the communication load and model accuracy of federated learning. From the structure of neural network models, it can be found that the data directly involved in determining the prediction results are the outputs of the last hidden layer. Due to this fact, we propose an assumption that the outputs of the last hidden layer of neural networks should be similar for samples with similar prediction results.

We take a classification task on the public Modified National Institute of Standards and Technology (MNIST) database as an example to verify the hypothesis. To facilitate presentation in the space-constrained paper[11], we set up a Multilayer Perceptron (MLP) with two hidden layers, each with 12 neurons. The learning rate is set to 0.01. Our experimental results are shown in Table 1. The first three rows of Table 1 are the results obtained during ten rounds of training, which record the dissimilarity of outputs of the last hidden layer under each predicted label. We denote the number of samples with label l by N_l. The dissimilarity of outputs of the k-th neuron can be formulated as:

$$\sigma_{l,k}^2 = \frac{1}{N_l}\sum_{n=1}^{N_l}(X_{l,k}^n - \bar{X}_{l,k})^2, \tag{1}$$

where $X_{l,k}^n$ refers to the output of the n-th sample with label l on the k-th neuron of the last hidden layer, and $\bar{X}_{l,k} = \sum_n X_{l,k}^n / N_l$. We take the samples with Labels 3, 6, and 7 in the dataset as examples, and similar results can be obtained under other labels. The last row of Table 1 is the dissimilarity of outputs of the last hidden layer blending all predicted labels. The 12 components represent the variance of the corresponding outputs of 12 neurons in the last hidden layer.

From the above experimental results, it can be found that the outputs of training samples with the same label in the last hidden layer are similar, because their variances are smaller than those in the scenario where we do not distinguish the labels. A similar phenomenon on the public dataset CIFAR-10 is also found. Accordingly, it can be considered that for neural network models on clients, the outputs of the last hidden layer can carry the characteristics of training samples to a certain extent.

3.2 Communication-Efficient Strategy Based on Model Split and Representation Aggregate

Based on the assumption in Subsection 3.1, we can conclude that for neural network models on different clients, an effect close to aggregating all parameters can be obtained by aggregating the outputs of the last hidden layer. The method of splitting neural networks has made great progress in split learning[23]. However, split learning requires too much communication between clients and the server due to communicating high-dimensional parameters. In terms of communication consumption per round, the communication load of split learning is not significantly reduced compared to federated learning. To adopt the paradigm of aggregating hidden layers in federated learning, this paper sets an output layer on the server side for complete training and retains a local model on each client side. As shown in Fig. 1, the server calculates the gradient of the loss function to the aggregated last hidden layer after local forward propagation, and distributes the gradient to clients. Each client uses this gradient to assist the update of its local model. Such learning process revises local models indirectly through back-propagation of gradients.

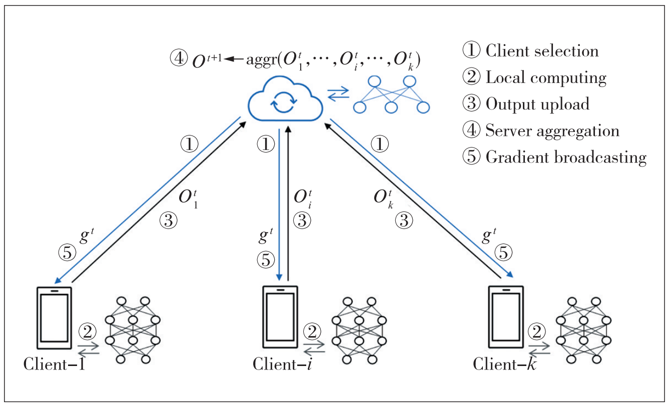
It is worth noting that we reform the neural network based federated learning on the server side and the client side respectively. Each client trains a local neural network, which together with the server constitutes a federated neural network framework as a whole. Unlike split learning, we keep the complete local model on each client and follow the setting of federated learning to support local training. In addition, the server does not collect local models or gradients uploaded by clients, which can reduce the risk of model increments or gradients compromising data privacy. Even malicious participants cannot launch inference attacks[24] or model inversion attacks[25] against honest clients by eavesdropping on local models. This means that our method could have stricter privacy and security, and in particular, has high communication efficiency. Each client participates in the aggregation process of federated learning by uploading the outputs of the last hidden layer, while the server coordinates a collaborative training by distributing the back-propagated gradient. Clients use this gradient to modify their local models after receiving the back-propagated gradient. More details are shown in Algorithm 1. We implement the federation of model training on the client and server through a local model training stage, a communication stage, and a back-propagation stage.

Algorithm 1. MSRA-Fed based on model split and representation aggregate

```
Require: The number of clients: $K$; learning rate: $\eta$; the number of local epochs: $E$; the number of global iterations: $T$; local datasets: $\{D_i\}_{i=1}^K$
Ensure: Local models on clients: $\{w_i\}_{i=1}^K$
//Server executes:
1:  Initialize: $w_i$ //Randomly initialize local models
2:  for each round $t = 1$ to $T$ do
3:      $m = \max(C \times K, 1)$
4:  $Z_t$ = random set of $m$ clients
5:  for each client $i \in Z_t$ in parallel do
```

▼Table 1. Variance of the outputs of each neuron in the last hidden layer

Label	1st Neuron	2nd Neuron	3rd Neuron	4th Neuron	5th Neuron	6th Neuron	7th Neuron	8th Neuron	9th Neuron	10th Neuron	11th Neuron	12th Neuron
Label-3	1.00	2.94	1.98	11.82	1.80	2.16	0.01	9.60	0.28	4.29	3.84	11.03
Label-6	4.82	0.00	3.98	1.88	0.00	3.65	1.17	0.03	36.54	8.05	3.65	0.72
Label-7	0.00	0.01	1.37	13.00	4.92	5.20	3.33	0.10	17.91	12.54	3.82	0.08
Blended	52.11	34.38	18.51	24.65	16.18	11.99	3.99	35.46	17.24	18.57	27.22	23.68



▲**Figure 1. Overview of MSRA-Fed**

```
6:        $(X_{i,l}, l) = \mathrm{ClientTrain}(i)$
7:    end for
8:    for each label $l$ do
9:        $X_l = \frac{1}{m}\sum_{i=1}^{m} X_{i,l}$
10:       Forward propagation with $(X_l, l)$, calculate $g$
11: end for
12:   for each client $i \in Z_t$ in parallel do
13:     ClientUpdate($i, g$)
14:   end for
15: end for
//Clients execute:
16: function ClientTrain($i$):
17: for each local epoch $e = 1$ to $E$ do
18:   for each sample $(s_n, l_n) \in D_i$ do
19:     if $e < E$ do
20:       $w_i \leftarrow w_i - \eta \cdot \nabla loss(w_i; s_n, l_n)$
21:     else do
22:       Forward propagation with $(s_n, l_n)$, calculate $X_l^n$
23:     end if
24:   end for
25: end for
26: for each label $l$ do
27:   $X_{i,l} = \frac{1}{N_l}\sum_{n=1}^{N_l} X_l^n$
28: end for
29: return a set of $(X_{i,l}, l)$
30:   function ClientUpdate($i, g$):
31:   Back propagation with $g$, update $w_i$
```

1) Local model training stage. As shown in Lines 17 – 28 of Algorithm 1, in each round of federated learning, each client trains a local model for several epochs and then uploads aggregated representation to the server for aggregation. The client preserves a full model locally instead of a split model. Unlike conventional federated learning, the client uploads the aggregated outputs of the last hidden layer instead of all parameters. In order to reduce communication consumption, the client adopts a class-wise average aggregation of the local hidden layer output set. The data that finally participate in global federated aggregation is the averaged hidden layer outputs under each label.

2) Communication stage. At Line 29 of Algorithm 1, the client communicates with the server. When the client uploads local data, the server collects the outputs of all clients' hidden layers and averages these local representations separately according to class (or label). Taking the classification task on MNIST as an example, the server will finally obtain the outputs of the last hidden layer corresponding to ten different labels after one communication.

3) Back-propagation stage. In Lines 8 – 14 of Algorithm 1, after averaging the hidden layers, the aggregated output is used as input to the neural network on the server to continue training and compute a gradient for back-propagation. After receiving the back-propagated gradient from the server, clients call it for back-propagation. That is, the back-propagated gradient of the federated averaged hidden layer is used to revise local models. For the client, the local training of the last epoch in each round is back-propagated using the gradient issued by the server side.

3.3 Summary

For the federated learning framework, we modify the collaborative training process and propose a communication optimization strategy based on model split and representation aggregation. Our method aggregates local representations on the server through the outputs of the last hidden layer, and uses gradients delivered by the server to coordinate training on each client. On the basis of the local training model, each client additionally introduces the server's gradient containing global information to correct its local model, which significantly reduces the communication load and also makes the model update more stable. In addition, since clients do not need to upload local models or gradients to any third party, the security risk of local data and models could be low. Although we did not focus on improving the accuracy of the model but on efficient communication, this will be improved in future work to sacrifice less accuracy under conditions of efficient communication and strict protection of privacy and security.

4 Evaluation

4.1 Dataset and Experimental Setup

We use the public dataset MNIST as the experimental dataset, which consists of ten classes of images. MNIST includes ten categories of handwritten digits, and each image is a grayscale image of size 28×28. This dataset contains 60 000 training samples and 10 000 test samples. Since the method designed in this paper is suitable for various neural network models, the commonly used model MLP is used here to verify the effectiveness of our method. Without loss of generality, we set up a learning environment with one central server and ten clients. The proportion of clients participating in each global training is $C = 1$, and the number of local epochs of the clients is $E = 5$. For the MNIST dataset, we employ an MLP with one hidden layer with 256 neurons. The learning rate η of MLP is set to 0.01. In the case of independent and identically distributed data, the samples are randomly shuffled. For each dataset used for training, all training samples are randomly and uniformly distributed to clients and each client randomly draws samples from it.

This experiment is carried out under Python 3.6, and the computer hardware is configured as 16 G memory, Intel i5-10400F CPU and GTX1650 GPU.

4.2 Evaluation Metrics

The communication efficiency and performance of the model is measured from the communication load required for model convergence and the final accuracy of the model. The definition of accuracy used here is shown in Eq. (2).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \qquad (2)$$

where TP is the number of true positives, which means that the positive samples are also predicted as positive by the model; TN is the number of true negatives, which means that the negative samples are also predicted to be negative; FP is the number of false positives, which means that the negative samples are predicted to be positive; FN is the number of false negatives, representing positive samples that are incorrectly predicted. The positive and negative here represent the true class and other classes of the sample, respectively.

4.3 Experimental Results and Analysis

Every time the federated learning progresses to the communication stage between the client and the server, we use an element in the tensor as a unit to count the amount of data transmitted in the communication. We define the communication load as the total amount of transmitted data. By comparing the communication load and the improvement percentage of model accuracy, or the communication load required to improve the model accuracy, the gap between the conventional federated neural network model and the improved communication-efficient MSRA-Fed can be clearly displayed.

Table 2 is a comparison of communication efficiency after five rounds of training on MNIST using MSRA-Fed and FedAvg. The value of communication load per 1% accuracy improvement indicates the average communication load consumed when the final trained model improves the accuracy by 1% compared to the initial model accuracy. The results show that although the initial accuracy of MSRA-Fed is lower than FedAvg by about 12%, the accuracy of two methods has reached more than 80% after five rounds of model training. Meanwhile, MSRA-Fed consumes less than 2% of the communication load of FedAvg.

The results after ten rounds of training on the MNIST dataset using MSRA-Fed and FedAvg are shown in Table 3. Both algorithms have almost converged after ten rounds of training. The results demonstrate that the communication load consumed by conventional FedAvg is 56.19 times that of MSRA-Fed when training for ten rounds. Moreover, the communication load of MSRA-Fed training for ten rounds is much lower than that of FedAvg training for five rounds, which shows that MSRA-Fed can be efficiently trained for more rounds in the same time and the accuracy improvement per communication load is high.

▼ **Table 2. Communication efficiency comparison after five rounds of training on MNIST dataset**

Method	Initial Accuracy/%	Accuracy After Five Rounds of Training/%	Communication Load/B	Communication Load per 1% Accuracy Improvement
FedAvg	71.62	86.10	6 352 000	54 834.25
MSRA-Fed	59.97	80.11	123 280	765.14

MNIST: Modified National Institute of Standards and Technology

▼ **Table 3. Communication efficiency comparison after ten rounds of training on MNIST dataset**

Method	Initial Accuracy/%	Accuracy After Ten Rounds of Training/%	Communication Load/B	Communication Load per 1% Accuracy Improvement
FedAvg	71.62	86.15	12 704 000	109 291.12
MSRA-Fed	59.97	80.48	226 080	1 337.86

MNIST: Modified National Institute of Standards and Technology

When we focus on "communication load per 1% accuracy improvement" in Tables 2 and 3, it can be seen that the strategy based on model split and representation aggregation requires far less communication load than FedAvg with the same accuracy improvement. This shows that our proposed method has much higher communication efficiency while sacrificing a little model accuracy. It is worth mentioning that the accuracy of our method is lower than that of FedAvg when the algorithm converges. This is because we only use the outputs of the last hidden layer instead of all parameters to aggregate global information in order to significantly improve the communication efficiency. Each client maintains a complete model locally, while introducing global information sent by the server during the training process to correct the local model. Compared to FedAvg, the accuracy of our method drops slightly, but remains within acceptable limits. Most importantly, we reduce the amount of communication significantly.

5 Conclusions and Future Work

In this paper, we modify the federated learning model for the problem of high communication load. Specifically, a method for efficient communication is designed based on model split and representation aggregation. By enabling the client to upload the outputs of the last hidden layer instead of all parameters and using the global information issued by the server to guide and correct the updates of each local model, the communication consumption of federated learning can be significantly reduced while ensuring the accuracy of the model. Furthermore, we experimentally validate the method proposed in this paper. The experimental results show that the model spit and representation aggregation mechanism can significantly reduce the required communication consumption, and the traditional training method FedAvg consumes 56.19 times the communication load than our method. While improving the communication efficiency, our method also guarantees the stability and accuracy of the model.

Backbone models other than neural networks are not explored in this paper, which will be our future work. This mechanism we design does not improve the accuracy of the model much when it converges, and may even have a slight negative impact. Future work will focus on addressing the above problems and exploring the possibility of large-scale application of our scheme in real environments.

References

[1] TALUKDER A, HAAS R. AIoT: AI meets IoT and web in smart healthcare [C]//13th ACM Web Science Conference 2021. ACM, 2021: 92 – 98. DOI: 10.1145/3462741.3466650

[2] ALKHATIB S, WAYCOTT J, BUCHANAN G, et al. Privacy and the Internet of Things (IoT) monitoring solutions for older adults: a review [J]. Studies in health technology and informatics, 2018, 252: 8 – 14

[3] LI T, SAHU A K, TALWALKAR A, et al. Federated learning: challenges, methods, and future directions [J]. IEEE signal processing magazine, 2020, 37 (3): 50 – 60. DOI: 10.1007/978-3-030-85559-8_13

[4] NGUYEN D C, DING M, PATHIRANA P N, et al. Federated learning for internet of things: a comprehensive survey [J]. IEEE communications surveys & tutorials, 2021, 23(3): 1622-1658. DOI: 10.1109/COMST.2021.3075439

[5] MCMAHAN B, MOORE E, RAMAGE D, et al. Communication-efficient learning of deep networks from decentralized data [C]//20th International Conference on Artificial Intelligence and Statistics (AISTATS). PMLR, 2017: 1273 – 1282. DOI: 10.48550/arXiv.1602.05629

[6] SHAHID O, POURIYEH S, PARIZI R M, et al. Communication efficiency in federated learning: Achievements and challenges [EB/OL]. (2021-07-23)[2022-05-01]. https://arxiv.org/abs/2107.10996v1

[7] CHAI Z, ALI A, ZAWAD S, et al. TiFL: a tier-based federated learning system [C]//29th International Symposium on High-Performance Parallel and Distributed Computing. ACM, 2020: 125 – 136. DOI: 10.1145/3369583.3392686

[8] TAN Y, LONG G, LIU L, et al. Fedproto: federated prototype learning across heterogeneous clients [C]//AAAI Conference on Artificial Intelligence. AAAI, 2022: 8432 – 8440. DOI: 10.1609/aaai.v36i8.20819

[9] THAPA C, CHAMIKARA M A P, CAMTEPE S, et al. Splitfed: when federated learning meets split learning [EB/OL]. (2022-02-16)[2022-05-01]. https://arxiv.org/abs/2004.12088

[10] KONEČNÝ J, MCMAHAN H B, RAMAGE D, et al. Federated optimization: distributed machine learning for on-device intelligence [EB/OL]. (2016-10-08)[2022-05-01]. https://arxiv.org/abs/1610.02527

[11] KHAN L U, SAAD W, HAN Z, et al. Federated learning for internet of things: recent advances, taxonomy, and open challenges [J]. IEEE communications surveys & tutorials, 2021, 23(3): 1759 – 1799. DOI: 10.1109/COMST.2021.3090430

[12] WANG L, WANG W, LI B. CMFL: mitigating communication overhead for federated learning [C]//IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2019: 954 – 964. DOI: 10.1109/ICDCS.2019.00099

[13] SATTLER F, WIEDEMANN S, MÜLLER K R, et al. Robust and communication-efficient federated learning from non-IID data [J]. IEEE transactions on neural networks and learning systems, 2019, 31(9): 3400 – 3413. DOI: 10.1109/TNNLS.2019.2944481

[14] KONEČNÝ J, MCMAHAN H B, YU F X, et al. Federated learning: strategies for improving communication efficiency [EB/OL]. (2017-10-30)[2022-05-01]. https://arxiv.org/abs/1610.05492

[15] SURESH A T, FELIX X Y, KUMAR S, et al. Distributed mean estimation with limited communication [C]//International conference on machine learning. PMLR, 2017: 3329 – 3337. DOI: 10.48550/arXiv.1611.00429

[16] CALDAS S, KONEČNY J, MCMAHAN H B, et al. Expanding the reach of federated learning by reducing client resource requirements [EB/OL]. (2019-01-08)[2022-05-01]. https://arxiv.org/abs/1812.07210

[17] DEAN J, CORRADO G, MONGA R, et al. Large scale distributed deep networks [C]//25th International Conference on Neural Information Processing Systems, NIPS. 2012: 1223 – 1231

[18] HUANG Y, CHENG Y, BAPNA A, et al. GPipe: efficient training of giant neural networks using pipeline parallelism [C]//33rd International Conference on Neural Information Processing Systems. NIPS, 2019: 103 – 112

[19] JIANG L, WANG Y, ZHENG W, et al. LSTMSPLIT: Effective SPLIT Learning based LSTM on Sequential Time-Series Data [EB/OL]. (2022-03-08)[2022-05-01]. https://arxiv.org/abs/2203.04305

[20] CHEN Y, SUN X, JIN Y. Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation [J]. IEEE transactions on neural networks and learning systems, 2019, 31 (10): 4229 – 4238. DOI: 10.1109/TNNLS.2019.2953131

[21] THAPA C, CHAMIKARA M A P, CAMTEPE S A. Advancements of federated learning towards privacy preservation: from federated learning to split learning [M]//Federated Learning Systems. Springer, Cham, 2021: 79 – 109

[22] WANG W, FENG F, HE X, et al. Denoising implicit feedback for recommendation [C]//14th ACM International Conference on Web Search and Data Mining. ACM, 2021: 373 – 381. DOI: 10.1145/3437963.3441800

[23] SINGH A, VEPAKOMMA P, GUPTA O, et al. Detailed comparison of communication efficiency of split learning and federated learning [EB/OL]. (2019-01-08)[2022-05-01]. https://arxiv.org/abs/1909.09145

[24] WAINAKH A, VENTOLA F, MÜßIG T, et al. User-level label leakage from gradients in federated learning [J]. Proceedings on privacy enhancing technologies, 2022(2): 227 – 244. DOI: 10.2478/popets-2022-0043

[25] KHOSRAVY M, NAKAMURA K, HIROSE Y, et al. Model inversion attack: Analysis under gray-box scenario on deep learning based face recognition system [J]. KSII transactions on internet and information systems (TIIS), 2021, 15 (3): 1100 – 1118. DOI: 10.3837/tiis.2021.03.015

Biographies

LIU Qinbo received his BS degree in mathematics and physics basic science from the School of Mathematical Sciences, University of Electronic Science and Technology of China in 2021. He is currently pursuing his ME degree with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. His research interests include federated learning and GNNs.

JIN Zhihao received his BE degree in computer science and technology from the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China in 2022. His research interests include federated learning.

WANG Jiabo received his BE degree in software engineering from the School of Information Science and Technology, Dalian Maritime University, China in 2021. He is currently pursuing his ME degree with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. His research interests include federated learning.

LIU Yang (liu.yang@hit.edu.cn) received his BE degree in computer science from the Ocean University of China, China in 2010, MSc degree in software engineering from Peking University, China in 2013, and DPhil (PhD) degree in computer science from the University of Oxford, UK in July 2018, advised by Prof. Andrew SIMPSON. He is currently an assistant professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. He is interested in security and privacy problems and, in particular, the privacy issues related to mobile and IoT devices.

LUO Wenjian received his BS and PhD degrees from the Department of Computer Science and Technology, University of Science and Technology of China, in 1998 and 2003, respectively. He is currently a professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. His current research interests include computational intelligence and applications, network security and data privacy, machine learning, and data mining. Dr. LUO is also a senior member of the Association for Computing Machinery (ACM) and the China Computer Federation (CCF). He has been a member of the organizational team of more than ten academic conferences, in various functions, such as the program chair, the symposium chair and the publicity chair. He also serves as the chair of the IEEE CIS ECTC Task Force on Artificial Immune Systems. He also serves as an associate editor or an editorial board member for several journals, including *Information Sciences, Swarm and Evolutionary Computation, Journal of Information Security and Applications, Applied Soft Computing*, and *Complex & Intelligent Systems*.