Received September 21, 2020, accepted November 1, 2020, date of publication November 16, 2020, date of current version December 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3038287

EdgeFed: Optimized Federated Learning Based on Edge Computing

YUNFAN YE[1], SHEN LI[1], FANG LIU[1], YONGHAO TANG[2], AND WANTING HU[3,4]
[1]School of Design, Hunan University, Changsha 410082, China
[2]School of Gifted Young, University of Science and Technology of China, Hefei 230026, China
[3]Canvard College, Beijing Technology and Business University, Beijing 100037, China
[4]China Academy of Electronics and Information Technology, Beijing 100041, China

Corresponding author: Fang Liu (fangl@hnu.edu.cn)

This work was supported in part by The National Natural Science Foundation of China under Grant 62072465, and in part by the National Science and Technology Major Project for Investigational New Drug (IND) under Grant 2018ZX09201-014.

ABSTRACT Federated learning (FL) has received considerable attention with the development of mobile internet technology, which is an emerging framework to train a deep learning model from decentralized data. Modern mobile devices often have access to rich but privacy-sensitive data, and computational abilities are often limited because of the hardware restriction. In previous works based on federated averaging (FedAvg) algorithm, mobile devices need to perform lots of calculations, and it is time-consuming in the process of global communication. Inspired by edge computing, we proposed edge federated learning (EdgeFed), which separates the process of updating the local model that is supposed to be completed independently by mobile devices. The outputs of mobile devices are aggregated in the edge server to improve the learning efficiency and decrease the global communication frequency. Empirical experiments demonstrate that our proposed EdgeFed has advantages in different bandwidth scenarios. Especially, by offloading part of the calculations from mobile clients to the edge server, the computational cost of the mobile devices and the global communication expense can be simultaneously reduced as compared to FedAvg.

INDEX TERMS Federated learning, deep learning, federated averaging, edge computing, edge federated learning.

I. INTRODUCTION

The development of mobile internet technology has promoted artificial intelligence to become a decisive force for human society to enter the intelligent era. Artificial intelligence has been gradually applied to life services that can be seen everywhere in social activities. For example, a survey of smart city monitoring [1] based on computer vision and data analysis, medical expert systems [2], [3] to assist doctors in diagnosis and the recommendation system [4] constructed by amounts of user behavior. Such application domains are all based on large amounts of data, data analysis and model training. Also, the increasing emphasis on privacy and security of personal information brings difficulties to the sharing and integration of data, becoming an obstacle to the sustained and healthy development of artificial intelligence. The proposal of FL [5] has partially solved the problem of isolated data islands. Many individuals or institutions have their private data and start updating model parameters through local training. Encrypted parameters are transferred in different parties to be aggregated into a better model, where all data are fully utilized while the privacy and security of data are ensured.

With the continuous development of mobile internet technology, mobile devices are becoming lighter and more portable, also with limited computational abilities and battery capacities due to the hardware restriction. In 2017, FedAvg [5] was proposed by Google, enabling local devices to train a model for several epochs, and transmit the trained parameters to the central server for aggregation. Local mobile devices in FedAvg play a main role in the process of whole training. However, mobile devices are not born for deep learning, too much continuous training will bring a negative impact on their sustainability. The possible heterogeneous communication conditions and computational abilities for different clients may also influence the global model aggregation. Consequently, computation offloading [6] is proposed to divide

The associate editor coordinating the review of this manuscript and approving it for publication was Utku Kose.

and offload complex computing tasks on mobile devices to the edge server for execution, taking advantage of the high bandwidth and low latency between clients and the edge server.

To combine the advantages of FL and computation offloading, in this paper, optimization algorithms for FL based on edge computing are proposed to implement the three-layer architecture of 'client-edge-cloud', the local updates of model parameters are performed on 'client-edge', and the global aggregation is between 'edge-cloud'. The primary benefits of EdgeFed are mainly as follows:

- The process of local updates for model parameters is divided into two parts, respectively on mobile devices and the edge server. Mobile devices can focus on the training of low layers, and more computational tasks are assigned to the edge server with richer computational resources. Thus, training needs in mobile devices become lighter and faster, which is more practical.
- The bandwidth between clients and the edge server is usually higher than it between the edge server and the central server, and the needed global communication frequency to reach a satisfying accuracy can be decreased by EdgeFed. Consequently, the global communication cost can be reduced compared with FedAvg.
- We explored the impact on the required global communication cost to train a desirable model under various bandwidth combinations, batch size and the number of epochs. Some qualitative guidelines were also given about picking key parameters to achieve better performance in both model accuracy and resources cost.

II. RELATED WORK

A. EDGE COMPUTING

Edge computing [7] is a new concept that has the potential to address the concerns of response time requirements and data privacy. Edge servers play an important role in edge computing by providing the required computing resources to achieve the objective of reducing latency. With the rapid development of edge computing, many research teams have explored related platforms. A specific edge computing platform ParaDrop [8] can provide computing and storage resources at the "extreme" edge of the network allowing third-party developers to flexibly create new types of services. A new concept of Cloudlet [9] is a trusted, resource-rich computer or cluster of computers that is well-connected to the Internet and available for use by nearby mobile devices. Also, there are some practical applications studied especially in driving, which is a typical field of high requirement for low latency. In [10], a collaborative learning setting on the edges was proposed, which can predict failures to ensure sustainable and reliable driving in a collaborative fashion. In [11], a prototype that leverages edge computing infrastructure was proposed, considering that driver's attention cost in vehicular applications should be scheduled for better interaction.

In edge computing, privacy security is gradually being valued by the public. To address potential location privacy and usage pattern privacy, a task offloading scheduling algorithm [12] was proposed based on the constrained Markov decision process. In [13], the differentially private mechanism was utilized to enable the training of deep convolutional neural networks (DNN) face recognition models to protect private data and model parameters. For privacy protection in home environments, a smart hub [14] is presented, which aims to empower users to determine how applications can access and process sensitive data collected by smart devices. An autonomous self-learning distributed system [15] utilized a federated learning approach for aggregating behavior profiles efficiently to detect intrusion. According to the potential data corruption on the edges due to external attacks or internal hardware failures, there are two protocols proposed in [16] to verify data integrity for mobile edge computing.

B. FEDERATED LEARNING

FL [5] is a collaborative machine learning framework that enables participating devices to periodically update model parameters while keeping all the training data on local devices. A survey [17] was conducted about resource allocation, data privacy and security in large-scale FL in the scenario of edge computing. An adaptive control algorithm [18] was proposed to determine the best trade-off between local update and global parameter aggregation to minimize the loss function under a given resource budget. In [19], the proposed extended FL protocol solves a client selection problem with resource constraints. A quasi-Newton method based vertical FL framework [20] for logistic regression under the additively homomorphic encryption scheme can considerably reduce the number of communication times. A comprehensive secure FL framework [21] was introduced including horizontal FL, vertical FL, and federated transfer learning to align the features and training samples between different data owners.

FedAvg [5] is a practical method for the FL of deep networks based on iterative model averaging, it is robust to the unbalanced and non-independent and identically distributed (non-IID) data. Based on FedAvg, many hierarchical FL works have emerged. A proactive content caching scheme [22] outperforms other learning-based caching algorithms without gathering private data based on a hierarchical FL architecture. Another hierarchical FL scheme [23] optimized the resource allocation among mobile devices to reduce communication latency in learning iterations. Even in the field of Electroencephalography classification [24], hierarchical FL was performed with increasing privacy-preserving demands. In [25], the proposed hierarchical architecture can reduce the model training time and the energy consumption of the end devices compared to cloud-based FL.

Works of previous hierarchical FL focus more on the interactions between various mobile devices and the edge server, whereas in the proposed EdgeFed, the process of model training is split into two parts. Model parameters of low-layers are learned in mobile devices, and those of high-layers are learned in the edge server, the consequent effects of key parameters are also investigated.

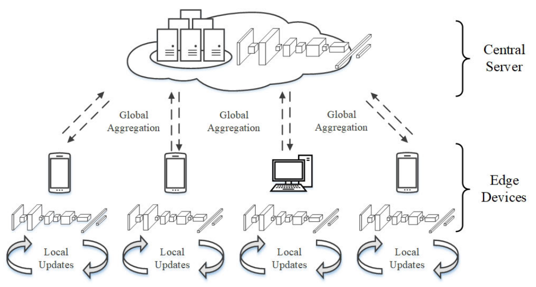


FIGURE 1. Structure of the FedAvg.

III. METHODS

A. STRUCTURE

The structure of the FedAvg is illustrated in Figure 1. There are several kinds of edge devices such as smartphones, laptops, surveillance cameras, etc., whose computational ability and storage capacity are limited. The central server is mainly deployed on the cloud and is composed of large amounts of computation and storage resources. In the FedAvg, when starting the federated learning, edge devices will first download a global model from the central server, training for several epochs with local data. Stochastic gradient descent (SDG) is utilized for optimization during local training. After several rounds of local updates, the results are uploaded to the edge server for global aggregation. Such a process is repeated until a model of desired performance is produced.

As we mentioned before, mobile devices often have a wealth of data but limited computational resources. Naturally, to make professional nodes just do the right things, most of the computational tasks can be offloaded to the edge server, which is relatively close to mobile devices and has sufficient resources. The structure of EdgeFed is shown in Figure 2.

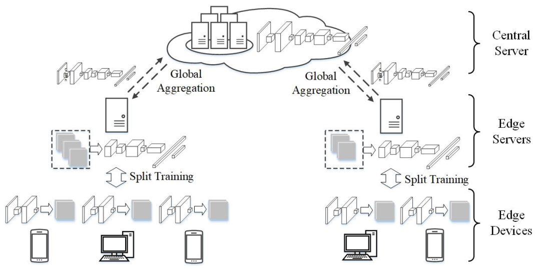


FIGURE 2. Structure of the EdgeFed.

There are mainly three parts in EdgeFed, which are edge devices, edge servers, and the central server.

Edge devices are intelligent end devices in the environment of the network edge, such as smartphones, cameras, smartwatches, AR/VR devices, smart vehicle platforms, smart glasses, etc. These edge devices are always good at data collection and data process but have limited battery capacity due to its characteristics of lightness and portability. The computational abilities of edge devices vary due to the differences in the basic hardware and system architecture. Also, data distribution and amount are often heterogeneous among different users in different application scenarios.

Edge servers are deployed in the environment of the network edge, such as routers, personal computers, micro base stations, or micro clouds in regional data centers. These edge servers are only one step away from edge devices, the data link is short and the bandwidth resources are sufficient, thus the speed of data transmission is relatively fast. Compared with edge devices, edge servers have more sufficient resources of computation and storage, as well as stable power supply.

The central server is a cluster of servers provided by the cloud service provider. It has rich storage resources and powerful abilities for data processing and analysis. However, since the central server lies in the top of the data link, which is distant to the network edge, also due to the instability of data transmission, a large amount of data transmission in the core network may cause network congestion.

In our proposed optimization algorithms, each iteration includes multiple split training between K edge devices and corresponding edge servers, and a global aggregation between m edge servers and the central server. Before the training, the central server will send the initial model parameters to multiple edge servers, and each edge device will download those parameters from the allocated edge server. The edge device k performs calculation with local data on low layers of the model. In our experiments, before the training of each batch, the raw data is processed through the first convolution layer and pooling layer, and the output of the pooling layer x_{pool}^{k}, will be sent to the edge server. After receiving the outputs of low layers from all edge devices, the edge server m will aggregate them into a larger matrix x_{pool}^{m}, which serves as the input of the remaining layers. The process can be represented as:

$$x_{pool}^{m} \leftarrow \left[x_{conv}^{1}, x_{conv}^{2}, \ldots, x_{conv}^{k}, \ldots, x_{conv}^{K}\right] \tag{1}$$

After the updating value is calculated on the edge server, the updated model parameters will be returned to edge devices. The calculation of model parameters can be explained as below, where ∇F is the loss function, g is the gradient, ω_t^i means model parameters in iteration i, and μ means the learning rate.

$$g = \nabla F(\omega_t) \tag{2}$$

$$\omega_{t+1}^{i} \leftarrow \omega_t^i + \mu g \tag{3}$$

After multiple split training between edge devices and the edge server, the edge server m will send the updated model parameters ω_t^i to the central server for the weighted average calculation to obtain the globally aggregated updated parameters ω_{t+1}^{i}, which will be returned to the edge server for the next iteration. The process can be calculated by:

$$\omega_{t+1}^{i} \leftarrow \sum_{m=1}^{M} \frac{n_m}{n} \omega_t^i \tag{4}$$

B. ALGORITHMS

According to the problem of high computational cost in mobile devices, we proposed a collaborative local update algorithm between the edge server and mobile devices after obtaining initial model parameters. The procedure is given in Algorithm 1.

Algorithm 1 Collaborative Local Update in Client-edge. There Are m Edge Servers; Every Edge Server Will Collaborate With k Clients; b Means the Local Batch Size; E Means the Number of Local Epochs; μ Means the Learning Rate

```
Require: local data from $k$ mobile devices
Ensure: updated model parameters
 1: //Run on edge servers
 2: procedure EdgeUpdate( ):
 3:     for each epoch $e$ from 1 to $E$ do
 4:         for each batch $b \in \beta$ do
 5:             for each client $k$ in parallel do
 6:                 $x^k_{conv}, y^k_{label} \leftarrow$ received from clien $k$
 7:             end for
 8:             $x_{conv} \leftarrow [x^1_{conv}, x^2_{conv}, \ldots, x^k_{conv}]$
 9:             $y_{label} \leftarrow [y^1_{label}, y^2_{label}, \ldots, y^k_{label}]$
10:             $\omega \leftarrow \omega - \mu \nabla loss(\omega; x_{output}, y_{label})$
11:         end for
12:     end for
13:     return $\omega$; // Send $\omega$ to cloud servers
14: end procedure
15:
16: //Calculation of low layers in clients
17: procedure ClientCompute($k, \omega$):
18:     for each epoch $i$ from 1 to $E$ do
19:         for each batch $b \in \beta$ do
20:             $x_{conv} \leftarrow Conv1AndPoolingLayer(\omega, b)$
21:             $send\ x_{conv}, y_{label}\ to\ EdgeServer$
22:         end for
23:     end for
24: end procedure
```

As shown in lines 17-24 of algorithm 1, after obtaining the initial model parameters, the mobile devices will start dividing local data into several batches of a fixed size and performing SGD. In each batch, the output from low layers of mobile devices and the corresponding data labels will be transferred to the edge server for aggregation. In the procedure of collaborative local updates, the process of every batch needs mobile devices to communicate with the edge server. Therefore, the value of batch size will influence the number of required local communications times t_{batch} between mobile devices and the edge server in a single training process for local data. As shown in equation 5, where D_k means the amount of local data in mobile device k and s_{batch} means the batch size.

$$t_{batch} = \frac{D_k}{s_{batch}} \tag{5}$$

When t is too large, which means there are too many communications between mobile devices and the edge server, the process of collaborative local updates may be influenced due to the instability of the network, even though there are high bandwidth and low latency. A small s_{batch} will increase the needed number of local communication times, whereas a large s_{batch} may influence the convergence of model training. Moreover, the aggregation on the edge server is synchronous in the process of collaborative local updates, namely, the edge server needs to wait for all participated mobile devices to complete calculations on low layers. Thus, all the number of local communication times t between mobile devices and the edge server should be the same. The total number of required communication times t_{local} between each mobile device and the edge server can be expressed as below, where E represents the number of local epochs.

$$t_{local} = E * \frac{D_k}{s_{batch}} \tag{6}$$

When E is too large, the total number of local communication times will increase. It is necessary to make trade-offs to choose the values of E and s_{batch}. Based on practical applications, we may as reasonably as possible decrease s_{batch} and increase E in the premise of an acceptable t_{local} for training the model better and faster.

After the process of collaborative local updates was completed by mobile devices and the edge server, multiple edge servers need to send the updated parameters to the central server, which will calculate the weighted arithmetic mean for aggregation. This process is given in Algorithm 2.

Algorithm 2 Global Aggregation. There Are m Edge Servers; Every Edge Server Will Collaborate With k Clients; b Means the Local Batch Size; E Means the Number of Local Epochs; μ Means the Learning Rate

```
Require: model parameters from each edge server
Ensure: global parameters after global aggregation
 1: //run on the central server
 2: procedure GlobalAggregation( ):
 3:     for each round $t = 1, 2, \ldots$ do
 4:         for each edge server $m$ in parallel do
 5:             $\omega^k_{t+1} \leftarrow EdgeUpdate(k, \omega_t)$
 6:         end for
 7:         //global aggregation based on parameters $\omega^k_{t+1}$
    from each edge server
 8:         $\omega_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n}{n_k} \omega^k_{t+1}$
 9:     end for
10: end procedure
```

After an iteration of global aggregation, the central server needs to determine whether it is necessary to continue the FL. The test dataset in the cloud data center can be used for testing the model. If it performs desirably, then there is no need to start the next iteration to avoid possible over-fitting and extra computation; whereas if the performance cannot satisfy the criterion, then the central server can repeat the whole process, distributing the model parameters to edge servers for the next round.

IV. EXPERIMENTS

In this section, the experimental environments, datasets, related models, experimental designs and results will be introduced. An overall comparison between the proposed EdgeFed and the FedAvg will be presented.

A. DESIGNS

We consider a hierarchical FL system with 100 mobile devices, 10 edge servers and a central server, assuming each edge server authorizes the same number of clients with the same amount of training data.

In the experiment of FedAvg, there are 100 clients and one central server participated. During one iteration, after downloading the initial model parameters from the central server, 100 mobile devices will train the model locally, each mobile device sends the locally updated model parameters to the central server synchronously for global aggregation. In the experiment of proposed EdgeFed, there are 100 mobile devices, 10 edge servers and one central server participated. During one iteration, after downloading the initial model parameters from the central server, 10 edge servers will distribute the parameters to allocated mobile devices, each edge server collaborates with 10 mobile devices. The training process of each batch starts with the calculation of low layers by mobile devices, the outputs will be sent to the corresponding edge server for aggregation and computation of subsequent layers, then the updated model parameters in the edge server will be returned to mobile devices for next batch. After multiple collaborative local updates are completed between mobile devices and the edge server, each edge server sends the local parameters to the central server for global aggregation, here is the end of one communication round.

Our experiment is based on the open-source dataset MNIST (Mixed National Institute of Standards and Technology database), which is a large database of handwritten digits collected by the National Institute of Standards and Technology, including 60,000 training images and 10,000 test images.

Since datasets in clients are always non-IID in practical FL scenarios, in the experiment of FedAvg, we first sort all the training data orderly according to the label of each image, then divide 60,000 images into 200 pieces, randomly distributing 2 pieces to each client. In the experiment of EdgeFed, we randomly select 6000 images for each edge server and sort them orderly based on the label, then they are divided into 20 pieces, every allocated client gets 2 pieces. Therefore, each mobile device has 600 training data with only part of class labels, the datasets among clients are non-IID.

SGD is used in the training process of both FedAvg and EdgeFed. The batch size and the number of epochs can not only influence the training efficiency, but also have a direct impact on the computational cost in mobile devices. Different batch sizes and numbers of epochs were employed for evaluating the performance of both FedAvg and EdgeFed. The main model of this experiment is convolutional neural network (CNN)-based, which consists of two convolutional layers with respectively 32 and 64 filters of size 5 × 5. Relu (Rectified Linear Unit) is used as the activation function, the size of two pooling layers is 2 × 2, and the last layer is a fully connected (FC) layer of 10 nodes. The structure of this model is represented in Figure 3.

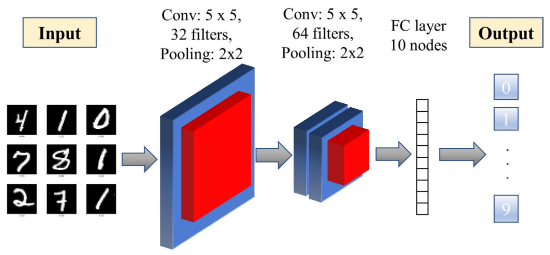

Input
Conv: 5 x 5, 32 filters, Pooling: 2x2
Conv: 5 x 5, 64 filters, Pooling: 2x2
FC layer 10 nodes
Output
0
1
9

FIGURE 3. Structure of the experimental model.

B. RESULTS AND ANALYSIS

At first, by setting different values of the batch size and the number of epochs, we compared the performance of the FedAvg and the EdgeFed, mainly about the accuracy and the training loss. The communication round means a complete training process including local updates in clients, local communications between clients and edge if in EdgeFed, and the global aggregation between clients/edges and the central server. The results are given in Figure 4 and Figure 5.

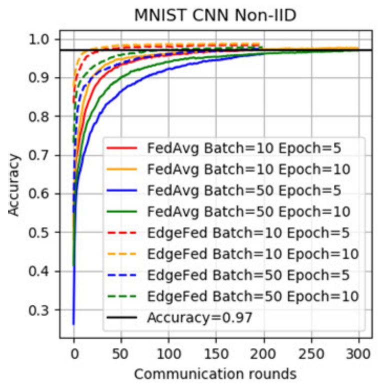


FIGURE 4. Accuracy vs. Communication rounds.

As shown in the results, the EdgeFed can achieve better training performance faster with a fewer number of global communication times. Since different batch sizes and numbers of epochs can determine the frequency of local updates, EdgeFed has a lower demand for computational cost in mobile devices than FedAvg under the same batch size and number of epochs. However, the process of local updates in EdgeFed should be accomplished collaboratively by mobile devices and the edge server, mobile devices should communicate with the edge server in every batch for training, uploading the results calculated in low layers to the edge server for aggregation and downloading the updated model parameters of high layers. Thus, model updates in each batch

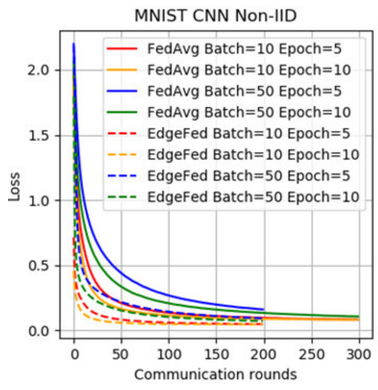


FIGURE 5. Loss vs. Communication rounds.

of EdgeFed will bring extra communication cost. The total communication cost T_{total} is composed of local communication cost T_{local} and communication cost of global aggregation T_{global}. Assume that between clients and the edge server, the total size of transmitted data is D_{local} bits, the bandwidth is B_l, and the data size is D_{global} between clients/edges and the central server with bandwidth B_g. In this case, the total communication cost can be calculated as:

$$T_{total} = T_{local} + T_{global} = t_{local} * \frac{D_{local}}{B_l} + t_{global} * \frac{D_{global}}{B_g} \tag{7}$$

Generally, the bandwidth between mobile devices and the edge server is smooth, whereas it is worse between clients/edges and the central server due to the long communication chain and possible congestions caused by the unstable core network. Therefore, referring to the transmission speed between the local area network and the wide area network, we assumed that the data transmission rate between mobile devices and the edge server ranges from 6MB/s to 10MB/s, and it is from 1MB/s to 5MB/s between clients/edges and the central server. Such assumptions are made to explore some qualitative guidelines on key parameters in different bandwidth scenarios. In our experiment, D_{local} is 793,168 bits in total, including the output of low layers with 125,440 bits, and the received model from the edge server with 667,728 bits. D_{global} is 1,335,456 bits in total, including model downloading and model uploading, each with 667,728 bits. Therefore, the time consumed in each batch and each round under different bandwidth can be calculated as $\frac{D_{local}}{B_l}$ and $\frac{D_{global}}{B_g}$, which are presented in Table 1. Meanwhile, we explored the communication status of FedAvg and EdgeFed when the model accuracy reaches 97% under different combinations of batch sizes and numbers of epochs. Table 2 demonstrates that, for both algorithms, the required times of global aggregation with batch size 50 is significantly higher than that with batch size 10, and it is even fewer when the number of epochs is 10. N_{local} means the required number of local communication times, and N_{global} is the required number of global communication times.

TABLE 1. Time consumed in each batch and each round.

	Global aggregation (667,728B+667,728B=1,335,456B)				
Bandwidth (MB/s)	1	2	3	4	5
FedAvg (ms/ round)	1273.6	636.8	424.5	318.4	254.7
EdgeFed (ms/round)	1273.6	636.8	424.5	318.4	254.7
	Local updates (125,440B+667,728B=793,168B)				
Bandwidth (MB/s)	6	7	8	9	10
EdgeFed (ms/batch)	126.1	108.1	94.6	84	75.6

TABLE 2. Required batches and rounds for reaching an accuracy of 97%.

		batch=10, epoch=5	batch=10, epoch=10	batch=50, epoch=5	batch=50, epoch=10
N_{local} (batches)	FedAvg	0	0	0	0
	EdgeFed	300	600	60	120
N_{global} (rounds)	FedAvg	200	150	320	300
	EdgeFed	30	15	130	90

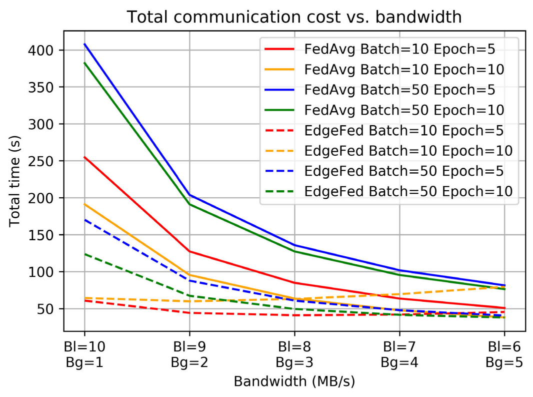


FIGURE 6. Total communication cost vs. bandwidth combinations.

Based on the data in Table 1 and Table 2, we can naturally get the T_{total} in various conditions, which are given in Figure 6. From left to right in the x-axis, the local bandwidth B_l and the global bandwidth B_g are getting closer with their sum fixed. An overall observation of the figure illustrates that the T_{total} to reach a certain test accuracy is a downward trend as we increase the B_g and decrease B_l in both FedAvg and EdgeFed. It is also noteworthy that, regardless of the orange lines, the T_{total} of EdgeFed is much lower than that of FedAvg, which demonstrates the advantage in total communication cost of the hierarchical FL. For the condition of $s_{batch} = 10$ and $E = 10$, the T_{total} decreases first and then increases as the value of the x-axis increases. Increasing B_g and decreasing B_l can reduce the total consumed time since the needed number of local communication times are often more than the times of global aggregation. If the B_g is much less than the B_l, then reducing the global communication frequency becomes the major factor to minimize T_{total}, whereas if the B_g is getting closer to the B_l, then reducing the sum of local and global communication times becomes the major factor. If the target is to minimize the total communication cost, we should carefully balance the batch size and epochs according to different bandwidth conditions by adjusting s_{batch} and E.

In the EdgeFed, less global aggregations are needed to reach a certain model accuracy, and mobile devices only need to train the parameters of low layers, while more data processing and weights updates are offloaded to the edge server. Thus, the computational requirement is greatly reduced, avoiding unnecessary energy consumption and resources occupation on mobile devices.

V. CONCLUSION

In this paper, we proposed optimization algorithms for FL based on edge computing to tackle the problem of large computational cost in mobile devices when performing FedAvg. We divided the process of local updates to completed by both mobile devices and the edge server, and the process of global aggregation is conducted between edge servers and the central server. The experiments show that the total communication and computational cost of mobile devices can be simultaneously reduced compared to FedAvg. Also, some guidelines were revealed about making trade-offs in selecting the value of key parameters in different bandwidth conditions.

REFERENCES

[1] R. Du, P. Santi, M. Xiao, A. V. Vasilakos, and C. Fischione, "The sensable city: A survey on the deployment and management for smart city monitoring," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1533–1560, 2nd Quart., 2019.

[2] S. Fang, Z. Cai, W. Sun, A. Liu, F. Liu, Z. Liang, and G. Wang, "Feature selection method based on class discriminative degree for intelligent medical diagnosis," *Comput., Mater. Continua*, vol. 55, no. 3, pp. 419–433, 2018.

[3] H. Chen, J. Yu, F. Liu, Z. Cai, and J. Xia, "Archipelago: A medical distributed storage system for interconnected health," *IEEE Internet Comput.*, vol. 24, no. 2, pp. 28–38, Mar. 2020.

[4] B. Smith and G. Linden, "Two decades of recommender systems at Amazon.com," *IEEE Internet Comput.*, vol. 21, no. 3, pp. 12–18, May 2017.

[5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[6] K. Akherfi, M. Gerndt, and H. Harroud, "Mobile cloud computing for computation offloading: Issues and challenges," *Appl. Comput. Informat.*, vol. 14, no. 1, pp. 1–16, Jan. 2018.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[8] P. Liu, D. Willis, and S. Banerjee, "ParaDrop: Enabling lightweight multi-tenancy at the Network's extreme edge," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2016, pp. 1–13.

[9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervas. Comput.*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[10] S. Lu, Y. Yao, and W. Shi, "Collaborative learning on the edges: A case study on connected vehicles," in *Proc. 2nd USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2019, pp. 1–8.

[11] K. Lee, J. Flinn, and B. D. Noble, "Gremlin: Scheduling interactions in vehicular computing," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 1–13.

[12] X. He, J. Liu, R. Jin, and H. Dai, "Privacy-aware offloading in mobile-edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.

[13] Y. Mao, S. Yi, Q. Li, J. Feng, F. Xu, and S. Zhong, "A privacy-preserving deep learning approach for face recognition with edge computing," in *Proc. USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2018, pp. 1–6.

[14] I. Zavalyshyn, N. O. Duarte, and N. Santos, "HomePad: A privacy-aware smart hub for home environments," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 58–73.

[15] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for IoT," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 756–767.

[16] W. Tong, B. Jiang, F. Xu, Q. Li, and S. Zhong, "Privacy-preserving data integrity verification in mobile edge computing," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 1007–1018.

[17] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.

[18] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[19] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[20] K. Yang, T. Fan, T. Chen, Y. Shi, and Q. Yang, "A quasi-Newton method based vertical federated learning framework for logistic regression," 2019, *arXiv:1912.00513*. [Online]. Available: http://arxiv.org/abs/1912.00513

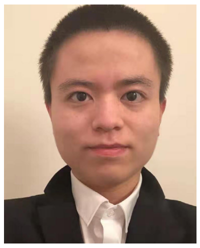
[21] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[22] Z. Yu, J. Hu, G. Min, H. Lu, Z. Zhao, H. Wang, and N. Georgalas, "Federated learning based proactive content caching in edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[23] M. S. H. Abad, E. Ozfatura, D. GUndUz, and O. Ercetin, "Hierarchical federated learning ACROSS heterogeneous cellular networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 8866–8870.

[24] D. Gao, C. Ju, X. Wei, Y. Liu, T. Chen, and Q. Yang, "HHHFL: Hierarchical heterogeneous horizontal federated learning for electroencephalography," 2019, *arXiv:1909.05784*. [Online]. Available: http://arxiv.org/abs/1909.05784

[25] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.



YUNFAN YE received the B.Sc. degree in computer science from Xiamen University and the M.Sc. degree in computer science from the Stevens Institute of Technology, USA, in 2019. His research interests include edge computing and computer vision.

SHEN LI received the B.Sc. degree from the University of Electronic Science and Technology of China and the M.Sc. degree from Sun Yat-sen University, in 2020. His research interests include edge computing and federated learning.

FANG LIU was born in 1976. She received the Ph.D. degree in computer science and technology from the National University of Defense Technology, in 2005. She is currently a Full Professor and a Ph.D. Supervisor with Hunan University. Her main research interests include computer architecture, edge computing, and storage systems.

YONGHAO TANG is currently pursuing the B.Sc. degree with the School of Gifted Young, University of Science and Technology of China. His research interests include edge computing and federated learning.

WANTING HU is currently pursuing the B.Sc. degree with the Canvard College, Beijing Technology and Business University. Her research interests include edge computing and federated learning.