Layered Model Aggregation based Federated Learning in Mobile Edge Networks

Qiming Cao[1], Xing Zhang[1]*, Yushun Zhang[1], Yongdong Zhu[2]
[1]Wireless Signal Processing and Network Laboratory
Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]Zhijiang Lab, Zhejiang Province, China
*E-mail: hszhang@bupt.edu.cn

2021 IEEE/CIC International Conference on Communications in China (ICCC) | 978-1-6654-4385-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCC52777.2021.9580403

***Abstract*—With the continuous improving of performance of the IoT and mobile devices, a new type of machine learning architecture, federated learning came into being. And there is also an increasing need to implement artificial intelligence frameworks on edge nodes. In this paper, we propose a federated learning system deployed in edge computing network, which realizes the server part in distributed form and uses layered model aggregation and dynamic topology to reduce bandwidth usage and time consuming. The experiment shows the effectiveness of federated learning algorithm in our system. And simulation results show that the time cost of our system increases logarithmically with the number of nodes rather than linearly in the traditional system.**

***Index Terms*—federated learning, edge computing**

I. Introduction

With the increasing number of end devices such as smartphones, wearable sensors and drones, huge amounts of data is generated at the edge of the network. However, due to limited wireless communication resources and privacy restrictions, it is inappropriate to transfer the training data from edge devices to servers, which makes traditional centralized machine learning face many difficulties. Therefore, a new branch of machine learning, federated learning, has emerged from the intersection of artificial intelligence and edge computing. Federated learning has inherent advantages in using edge computing to process data, but there are also many challenges, such as the large amount of uplink bandwidth required for the terminal to upload the training results, the insufficient computing capacity of the terminal and so on.

In view of the advantages and challenges of federated learning, this paper deploys federated learning in edge computing networks with the model aggregation process gradually completed on different edge computing nodes, which can decentralize the computing tasks to each node as much as possible and reduce the uplink bandwidth usage.

The paper is organized as follows. In section 2, we describe the related work. In section 3, the framework of our system is presented. The simulation and analysis are shown in section 4. Finally in section 5, we provide a conclusion.

This work is supported by the National Science Foundation of China under Grant 61631005 and by the Zhijiang Laboratory Open Project Fund 2020LCOAB01.

II. Related Work

In this section, we will introduce traditional system and previous studies in federated learning.

A. Traditional Federated Learning System

In the basic architecture there is one server and some end devices. They collaboratively learn a machine learning model. In a round of federated learning, the end-user devices download the current model from the server, and perform local training with their local data. The server then collects the updates from the end-user devices and aggregates them to get new model [1].

B. Previous Studies

There are extensive studies on the application of federated learning, such as vehicular networks [2], healthcare [3], and mobile keyboard prediction [4]. Current researches on horizontal federated learning are mainly focused on user-server architecture, and most federated learning application scenarios of mobile terminals are suitable for this classic architecture. For example, Google utilizes federated learning in mobile terminals such as mobile phones and tablets to improve google keyboard query suggestions [5].

Due to the limited computing capacity and battery of mobile devices and unstable wireless network connection, studies in this field focus on how to select appropriate mobile terminals to participate in training [6] [7] [8], and how to balance the power consumption of the mobile terminal with the cost of wireless transmission power or local computing [9]. Many researches focus on allowing a large number of learning nodes to collaborate on a shared global neural network model [10] [11], while other researches [12] [13] [14] focus on gradient compression in order to ensure the privacy and security of user data and overcome the communication bottleneck in federated learning.

But as the number of end devices involved in federated learning increases, the method mentioned above is not enough to solve the bandwidth shortage. Therefore, we propose a scheme to deploy federated learning in edge computing networks in this paper, which uses edge computing to aggregate model layer by layerand reduce redundant transmission.

III. System Descriptions and Models

In this section, we will introduce the main components and workflow of the system, and the algorithms used in this system are illustrated as well.

A. System Descriptions

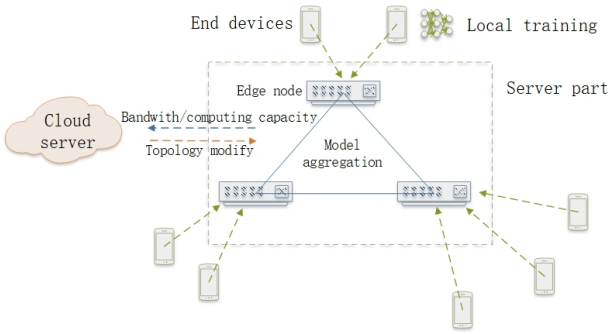

Fig. 1. system structure of federated learning

The system is mainly composed of two parts, several edge computing nodes and a cloud service. The cloud service does not participate in the federated learning process, and is mainly responsible for guiding the end devices to connect to the appropriate edge node, controlling the joining and exiting of edge nodes and monitoring the working state of them. In order to carry out the model aggregation progress in federated learning, a tree-shaped topology is formed between edge computing nodes. The edge computing nodes located at the root node of the structure is called root node, where the results of each round of federated learning are finally aggregated to the root node to get new model. In our system the cloud service modifies the topology of the edge nodes based on the real-time bandwidth occupancy of the links between them and estimated local training time.

According to different application scenarios, local model training can be performed by each edge node or by end devices connected to the edge node (depending on where the sensitive data is). In the latter situation, the structure of each edge node and its end devices is similar to the classic server-user architecture. Since this paper mainly discusses the implementation of the server part, we treat edge nodes and end devices as a whole. For example,in the latter situation, the local training time of edge node is refer to the time that the edge node waits for its end devices to complete the local training and upload the results.

To some extent, the main difference between this system and the basic one is that it realizes the server part using distributed edge computing network rather than a single node.

B. The Way to Aggregate Model Layer by Layer

Compare with the basic architecture, ours has many edge nodes that can be used in model aggregation. This paper proposes an aggregation method using the topology of the edge nodes to progress model aggregation layer by layer. This aggregation method distributes the computing tasks of the root node to the aggregation nodes of each layer, reducing the computing pressure of the root node and the redundancy of data transmission in the network greatly. In the basic architecture, all the end devices connect to the same server, resulting in congestion. Since the aggregation has been performed before transmission, the layered and stepwise aggregation method can ensure that only one gradient data needs to be transmitted in a link at most.

The steps of edge nodes aggregation process are as follows:

1) Complete the local training.
2) If have child nodes, wait for the child nodes uploading training results.
3) Aggregate the local training results with the child nodes'.
4) Upload the aggregated results to the upper node.

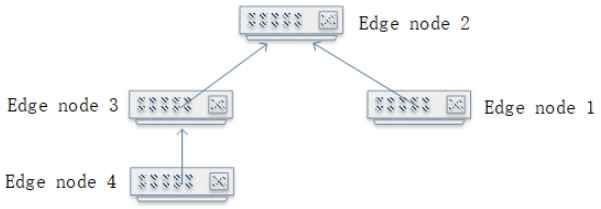

Fig. 2. topological structure of federated learning

For example, ω^i denotes the local training result of edge i and ρ_i denotes the weight of edge i, which can be set as the amount of end devices connected to the edge node. In the topology shown in Figure 2, edge nodes 1 and 4, which has no child node, start uploading data firstly. Edge node 1 multiplies the parameters of the local model ω^1 with local weight ρ_1, and uploads $\rho_1\omega^1$ and ρ_1 to edge node 2. And edge node 4 uploads $\rho_4\omega^4$ and ρ_4 to edge node 3 at the same time. Edge node 3 performs an aggregation operation and uploads $\rho_3\omega^3 + \rho_4\omega^4$ and $\rho_3 + \rho_4$ to edge node 2. Edge node 2 now have received data from all child nodes, and calculates $(\rho_1\omega^1+\rho_2\omega^3+\rho_3\omega^3+\rho_4\omega^4)/(\rho_1+\rho_2+\rho_3+\rho_4)$ as the result.

C. Federated Learning Algorithm Based on FSVRG and ADAM

FSVRG algorithm [15] with small changes is adopted in this system. Here's how to use the algorithm in our system.

We assume that there are m edge nodes and each edge node has n samples $z^1, \cdots, z^n$, and f is the loss function for each sample where MSE can be used or others chosed as need. Our goal is to minimize the overall loss function $F(\omega)$.

$$F(\omega) = \frac{1}{m}\sum_{i=1}^{m} F_i(\omega) \tag{1}$$

$$F_i(\omega) = E[f(\omega, z)] = \frac{1}{n}\sum_{j=1}^{n} f(\omega, z_i^j) \tag{2}$$

$$\hat{\omega} = argminF(\omega) \tag{3}$$

In the t-round iteration, m edge nodes locally calculate $\nabla F_k(\omega^t)$. After aggregation layer by layer the root node gets

global gradient $\nabla F(\omega^t)$. Then $\nabla F(\omega^t)$ is broadcast to each edge node. And each edge node solves (4) locally.

$$\omega_k = \underset{\omega \in R^d}{argmin}\, F_k(\omega) - \left(\nabla F_k(\omega^t) - \eta \nabla F(\omega^t)\right)^T \omega \tag{4}$$

The stochastic gradient descent method(SGD) can be used to solve this problem, in which each node randomly selects a sample from the local data to iterate (5) several rounds.

$$\omega_k = \omega_k - h(\nabla F_i\left(\omega_k\right) - \nabla F_i\left(\omega^t\right) + \nabla \mathrm{F}\left(\omega^t\right)) \tag{5}$$

The root node also gets g_t after aggregation layer by layer according to (6), and use ADAM [17]((7) to (9)) to get new model parameters $\omega^{(t+1)}$. Note that m_t and v_t should be corrected as (10) when the first time using ADAM.

$$g_t = \frac{1}{m}\sum_{k=1}^{m}\left(\omega^t - \omega_k\right) \tag{6}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{7}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{8}$$

$$\omega^{t+1} = \omega^t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{9}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1}, \hat{v}_t = \frac{v_t}{1-\beta_2} \tag{10}$$

It can be seen that there are two rounds of aggregation between each node in one round fedrated learning. In the first round the global gradient is obtained, while global gradient and SGD are used to solve the problem (4) in the second round. ADAM is also used, which means the root node in the second round need to store the history data m_t and v_t. If the root node is changed during two rounds of federated learning, the history data should be transferred from old root node to the new one.

D. A Heuristic Topological Decision Algorithm in Fedrated Learning

During different rounds of model aggregation layer by layer, the topology of edge nodes in fedrated learning can be changed according to the CPU usage in each node and network bandwidth usage between them. In order to find an appropriate topology, we need to find a method estimating the time cost by a round of fedrated learning using different topologies. In general, local training time or parameters transmission time is much lager then the aggregation time, so we ignore the aggregation time here. To further illustrate the optimization goal, the following statements are made:

- Each node does not receive data from its child nodes until it completes its local training.
- Only after receiving data from all child nodes can a node start transmitting data to its parent node.
- A node receives data from only one child node at a time.

Now we can estimate the time of a round of fedrated learning with algorithm 1. In algorithm 1, $trans\_t$ is a $n \times n$ matrix, where $trans\_t\,[i]\,[j]$ is the transmission time from node i to node j, and $lcoal\_t\,[i]$ is the local training time of node i. And the time of node i finishing receiving data from all child nodes is denoted as $completion\_t\,[i]$.

Algorithm 1 An algorithm for estimating completion time

Input: topology, $trans\_t$, $lcoal\_t\,[i]$
Output: $completion\_t$ of root node

1. From the nodes that have not been assigned $completion\_t$ yet, find all the child nodes that have been assigned $completion\_t$.
2. For each node i found in 1,do
   (a)$completion\_t\,[i] \leftarrow local\_t[i]$
   (b)Sort the child nodes of i from smallest to largest by $completion\_t\,[j]$, for each child node j do:
   if $completion\_t\,[i] > completion\_t\,[j]$
   $completion\_t\,[i] \leftarrow completion\_t\,[i] + trans\_t\,[j]\,[i]$
   else
   $completion\_t\,[i] \leftarrow completion\_t\,[j] + trans\_t\,[j]\,[i]$
3. **if** $completion\_t$ of root node is not assigned yet, go back to step 1.

At first glance, this problem looks like the minimum spanning tree problem, but it's actually much more complicated. First of all, the upstream and downstream bandwidth is not symmetric, and which node is the root node directly affects the overall completion time,so the traditional algorithm of minimum spanning tree is not applicable. Since a node may have to wait other nodes finishing computing or transmission, it is difficult to get a closed solution. According to Caylay formula, the number of free spanning tree of fully connected graph with n different nodes is n^{n-2}, consider the different root node, there are n^{n-1} different topologies. As the number of nodes increases, the number of possible topologies increases dramatically, so it is clear that enumeration methods do not work either. Therefore, in order to solve this problem, we design a heuristic algorithm based on the following intuitive ideas:

- The nodes with high computing capacity should be placed at a deeper location, so the nodes with low computing capacity can have more sufficient time to complete the local training.
- Let the completion time of each child node not equal, such as an arithmetic sequence, to avoid waiting as much as possible.

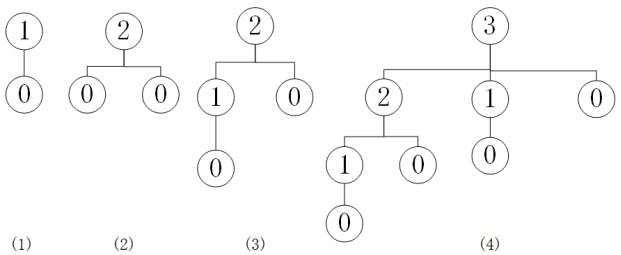

Fig. 3. illustration of saturated structure

Consider an extreme case that all nodes have the same local training time and the same bandwidth across them. Since all the nodes start local training at the same time, we can consider only the transmission time. Figure 3-(1) shows the topology of two nodes, and the number in the node is the completion time of the node. Since the local training time is not taken into account, the completion time of the child node is 0. And the root node has to wait for the child node transmitting data, which is denoted as 1 here for one data transmission time. When the number of nodes increases to 3, the completion time of root node increases to 2. In Figure 3-(2), both child nodes have 0 completion time but since one child node has to wait for the other one transmitting data. Increasing the completion time of a child node to 1 does not affect the completion time of the root node, so one more node is add to the child node in Figure 3-(3), and it's impossible to add more node while keeping root node's completion time unchanged. We call this a saturated structure. The saturated structure are showed in Figure 3-(1),(3),(4). The completion time of the child nodes of each node in the saturated structure is an arithmetic sequence. The saturated structure with m completion time is equivalent to the saturated structure containing 0 to $m-1$ respectively under the root node, and likewise, the saturated structure with $m-1$ completion time is equivalent to the saturated structure containing 0 to $m-2$ respectively under the root node. Therefore, it can be seen that the number of nodes in saturated structure with m completion time is twice as much as it with $m-1$ completion time. It is not difficult to conclude that the saturated structure of completion time m has 2^m nodes, which means that when the number of nodes $n = 2^m$, only the saturated structure with completion time m is the best. When $2^{m-1} < n < 2^m$, the completion time is at least m, but with many topologies satisfied. One possible strategy is to remove several subtrees from root node of saturated structure with completion time m.

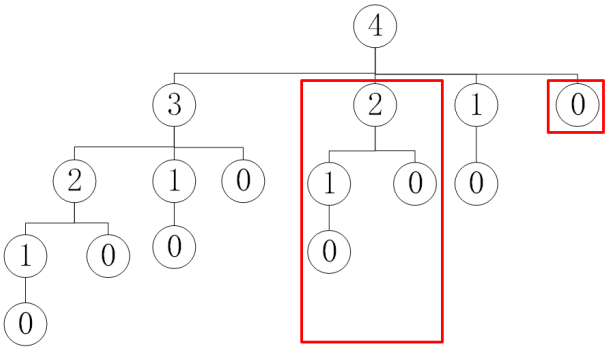

Fig. 4. illustration of saturated structure

For example, when $n = 11$, $2^3 < n < 2^4$, we should remove $2^4 - n = 5$ nodes in Figure 4. Since $5 = 2^0 + 2^2$, the removed nodes can be the subtrees of 0 and 2 completion time which are marked by red boxes. We call the topology left as saturated structure of n nodes. Once the topology is obtained, each node can be placed into the structure according to a certain strategy. Now we have algorithm 2. In algorithm 2, n denotes the number of nodes, $saturation\_m[i]$ denotes the number of nodes whose completion time is i in the extreme case mentioned above. And $trees[i](i \in \{0, 1, \ldots, m\})$ are sets to store these nodes.

Algorithm 2 A heuristic topological decision algorithm

Input: n, $trans\_t$, $lcoal\_t\,[i]$

Output: $trees[m]$

1. Find m let $2^{m-1} < n \leq 2^m$, saturation_m[m]$\leftarrow$ 1,saturation_m[i]$\leftarrow 2^{m-i-1}(i \in \{0, \ldots, m-1\})$, Initialize $trees[i](i \in \{0, \ldots, m\})$ to be empty.
2. For every bits of $2^m - n$ (unsigned int): if (i+1)-th bit is 1, then $saturation\_m[i] \leftarrow saturation\_m[i] - 1$ and $saturation\_m[j] \leftarrow saturation\_m[j] - 2^{i-j-1}(j \in \{0, \ldots, i-1\})$
3. Sort the child nodes from smallest to largest by $local\_t$ and put them into a queue named $unused\_node$. Let set $trees[i](i \in \{0, \ldots, m\})$ empty
4. For each i in $\{0, 1, \ldots, m\}$ do $saturation\_m[i]$ times:
   (a)Take a node from $unused\_node$(whos index denoted as r) as the root node of a new subtree
   (b)**if** i is not 0, for j in $\{0, 1, \ldots, i-1\}$ **do**
   For each node in $trees[j]$, use algorithm 1 to estimate $completion\_t[r]$ after adding the node as child node. Find out one leading to the smallest $completion\_t[r]$.
   Add that node as root node's child node .
   Remove that node from trees[j] and update $completion\_t[r]$.
   (c)Put node r in trees[i]

E. The Process by which the Cloud Service Modifies the Topology

In a specific application the model trained by federated learning is predetermined, so as well the size of parameters and the calculation needed for local training. Therefore, the cloud server can estimate the transmission time and local training time through residual computing capacity of each nodes and unused bandwidth between them. With the transmission time and local training time, the cloud server is able to make a strategy for edge nodes to modify current topology. The main steps are as follows:

1) Read the residual computing capacity of each nodes and unused bandwidth beetween them.
2) Estimate the transimition time and local traning time.
3) Use algorithm 2 to get the new topology.
4) If current aggregation progress is the second round of federated learning algorithm described in part C, use algorithm 1 get completion time of new topology and old topology. If the latter is less than the former plus twice the transition time from the old root to the new root, keep current topology unchanged and exit, otherwise take step 5.
5) Send and apply new topology to each edge node.

IV. Performace Simulation and Analysis

A. Experiments on Federated Learning Algorithm

In order to verify the validity of the algorithm and method used in our federated learning system, we deployed the system on four edge nodes and conducted experiments based on mnist handwritten datasets [16]. Four edge nodes contain non-iid data. Specifically, node 1 has 400 data of tag 0, 400 data of tag 1, and 200 data of tag 2. Node 2 has 200 data of tag 2, 400 data of tag 3, and 400 data of tag 4. Node 3 has 400 data of tag 5, 400 data of tag 6, and 200 data of tag 7. Node 4 has 200 data of tag 7, 400 data of tag 8, and 400 data of tag 9. The model used in the experiment is a simple convolutional neural network, which consists of two convolutional layers, a full connection layer and a softmax layer.

The experimental results are shown in the Figure 5. Obviously, the convergence speed and accuracy of FSVRG with ADAM algorithm performs better than simple FSVRG algorithm or SGD algorithm, which demonstrates the effectiveness of the new algorithm in our federated learning system.

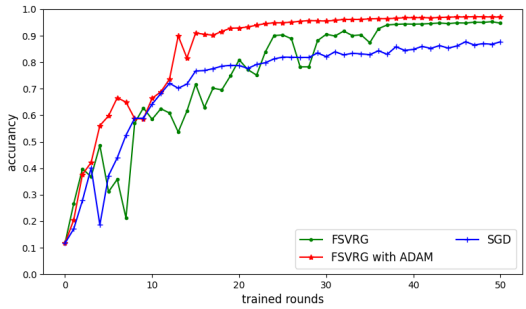

Fig. 5. comparison of federated learning algorithm

B. Experiments on Dynamic Topology

The experiment of dynamic topology structure adopts the form of simulation experiment. Given the number of nodes, the time required for local training of each node and the time required for data transmission between each node(the transmission time reflects bandwidth limitations) are randomly generated. The topology structure under the current parameters is obtained by the dynamic topology decision algorithm proposed in this paper, and the structure is compared with some other structures.

In Figure 6 it can be seen that the average time required for a round of federated learning with star shaped topology (the traditional user-server structure) is larger then the others, especially when there are a large number of nodes. The main reason is that the federated learning of the star structure is limited by the server bandwidth. When the number of nodes increases to a certain extent, the time required for aggregation will increase linearly with the number of users. This also reflects the advantages of layered aggregation, which means different aggregation nodes are parallel to each other, making

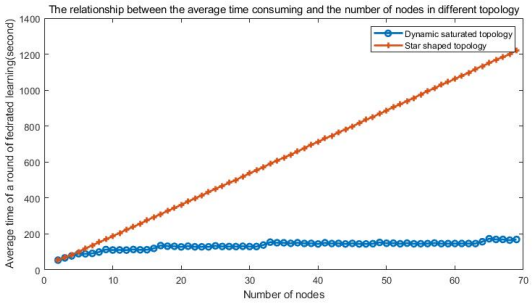

Fig. 6. comparison with traditional user-server structure

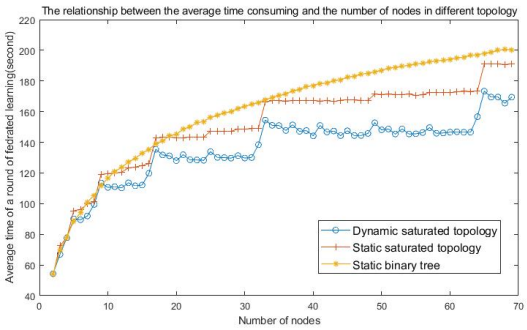

Fig. 7. comparison with binary tree and static saturated structure

full use of network bandwidth and reducing unnecessary transfers.

In the Figure 7, the static saturation structure refers to the saturation structure mentioned above but does not change with the bandwidth between each node. The binary tree in the figure above is also a static binary tree structure. It can be seen that the average time of the saturated structure increases in a step-like way. This is because when the number of nodes reaches 2^m, the depth of the saturated structure increases by 1, and the average time increases by about one time required for data transmission. However, when n is between 2^{m-1} and 2^m, the depth of saturated structure is constant, so the change of average time is not obvious. It can be seen from the figure 7 that the dynamic saturated structure is obviously better than the static saturated structure and the binary tree structure.

If we only look at the situation of $n = 2^m$, we can see that the average time increases logarithmically with the number of nodes. That's because when n doubled, m increases 1, and the depth of the tree structure increases 1, so averagely the completion time increases average transmission time.

V. Conclusion

In this paper, we propose a federated learning system deployed in edge computing network and explain the workflow and advantages of our system. For details, we also present the federated learning method based on FSVRG and ADAM and the topology decision algorithm to help building the federated learning system. The experiment and simulation demonstrate the efficiency of our system.

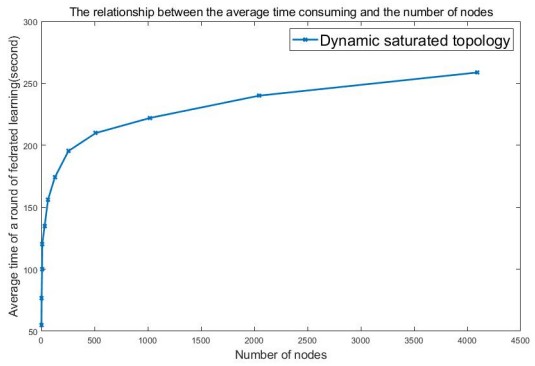

Fig. 8. the average time increases logarithmically with the number of nodes

REFERENCES

[1] McMahan B, Ramage DFederated learning :Collaborative machine learning without centralized training data [J/OL][2018-12-23]. https://www.googblogs.com/federated-learning-collaborative-machine-learning-without-centralized-training-data/

[2] Samarakoon S , Bennis M , Saad W , et al. Federated Learning for Ultra-Reliable Low-Latency V2V Communications. IEEE, 2018. I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[3] Xu J, Glicksberg B S, Su C, et al. Federated learning for healthcare informatics[J]. Journal of Healthcare Informatics Research, 2021, 5(1): 1-19.

[4] Hard A, Rao K, Mathews R, et al. Federated learning for mobile keyboard prediction[J]. arXiv preprint arXiv:1811.03604, 2018.

[5] Yang T, Andrew G, Eichner H, et al. Applied federated learning: Improving google keyboard query suggestions[J]. arXiv preprint arXiv:1812.02903, 2018.

[6] Nishio T , Yonetani R . Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge[J]. 2018.

[7] Yang H H , Arafa A , Quek T Q S , et al. Age-Based Scheduling Policy for Federated Learning in Mobile Edge Networks[J]. 2019.

[8] Yang, Howard H., et al. "Scheduling policies for federated learning in wireless networks." IEEE Transactions on Communications 68.1 (2019): 317-333.

[9] Tran N H , Bao W , Zomaya A , et al. Federated Learning over Wireless Networks: Optimization Model Design and Analysis[C]// IEEE INFOCOM 2019 - IEEE Conference on Computer Communications. IEEE, 2019.

[10] Abadi M, Agarwal A, Barham P, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems[J]. arXiv preprint arXiv:1603.04467, 2016.

[11] Li M, Andersen D G, Park J W, et al. Scaling distributed machine learning with the parameter server[C]//11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14). 2014: 583-598.

[12] Lin Y, Han S, Mao H, et al. Deep gradient compression: Reducing the communication bandwidth for distributed training[J]. arXiv preprint arXiv:1712.01887, 2017.

[13] Konečný J, McMahan H B, Yu F X, et al. Federated learning: Strategies for improving communication efficiency[J]. arXiv preprint arXiv:1610.05492, 2016.

[14] Dryden N, Moon T, Jacobs S A, et al. Communication quantization for data-parallel training of deep neural networks[C]//2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC). IEEE, 2016: 1-8.

[15] Konečný J, McMahan H B, Ramage D, et al. Federated optimization: Distributed machine learning for on-device intelligence[J]. arXiv preprint arXiv:1610.02527, 2016.

[16] Mnist handwritten datasets: http://yann.lecun.com/exdb/mnist.

[17] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[18] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang and W. Wang, "A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications," in IEEE Access, vol. 5, pp. 6757-6779, 2017, doi: 10.1109/ACCESS.2017.2685434.