RESEARCH ARTICLE

Decentralized Federated Meta-Learning Framework for Few-Shot Multi-Task Learning

Xiaoli Li[1] | Yuzheng Li[1] | Jining Wang[1] | Chuan Chen[1] | Liu Yang[2] | Zibin Zheng[1]

[1]the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, 510000, China

[2]the School of Computer Science and Technology, Tianjin University, Tianjin, 30000, China

Correspondence
Chuan Chen, the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, 510000, China
Email: chenchuan@mail.sysu.edu.cn

Funding information
The research is supported by the National Key R&D Program of China (2020YFB1006001), the National Natural Science Foundation of China (62176269, 62076179, 61732011), the Guangdong Basic and Applied Basic Research Foundation (2019A1515011043), the Beijing Natural Science Foundation (Z180006) and the Innovative Research Foundation of Ship General Performance (25622112).

Federated Learning is increasingly attractive, however as the number of training samples on a single device is too small and the training tasks of the devices are different, it faces the Few-Shot Multi-Task Learning problem. Moreover, federated learning frameworks are usually vulnerable to malicious attacks of the central server and diverse clients. To address these problems, we propose a Decentralized Federated Meta-Learning Framework for few-shot multi-task learning (DFMLF). In DFMLF, the devices take the rapid adaptation as objective and learn the meta-knowledge shared by tasks to deal with the few-shot multi-task problem. In addition, DFMLF conducts cross-validation and secure aggregation mechanism by a small number of committee nodes, which not only eliminates the central server to avoid the security risks brought by the malicious central server, but also avoids the attack of malicious devices. Moreover, to address the extra communication cost brought by the committee strategy, we propose a communication-efficient method to make the training and aggregation carried out in parallel. We conduct extensive experiments based on real-world datasets, and the experimental results demonstrate the ef-

Abbreviations: Decentralized Federated Meta-Learning Framework for Few-Shot Multi-Task Learning.

fectiveness, robustness, and efficiency of our framework.

KEYWORDS
Federated Learning, Meta-Learning, Committee, Few-Shot, Multi-Task

1 | INTRODUCTION

In recent years, machine learning has penetrated all aspects of our lives. A practical and deployable machine learning algorithm often requires a huge amount of high-quality data for training. However, in many real scenes, the data are scattered in different places. For example, in mobile computing, smart mobile devices such as smartphones, mobile sensors, wearable devices, and autonomous vehicles are distributed in different places and generate a tremendous amount of valuable data. The conventional machine learning methods usually require integrating the distributed data into a central server. However, due to the consideration of data security and user privacy, this is usually impractical, which leads to the problem of data island [1].

Recently, federated learning [2, 3] has been proposed to solve the data island problem, in which the devices can train a shared global model collaboratively without exposing their data to others, and the performance of the global model is very close to that of the model based on the fused data. Federated Averaging algorithm (FedAvg) [4] is the most widely recognized federated learning method, which usually has one central server and multiple clients, and contains three steps in each communication iteration: (1) the server sends the current global model to some selected clients; (2) the selected clients receive the global model, perform training based on their local data, and then send the updated local models to the server; (3) the server aggregates the uploaded local models of clients to generate a new global model by averaging. The process is repeated until it converges. Federated learning provides globally model training while ensuring privacy, thus is increasingly attractive.

However, federated learning on mobile devices is not a trivial task. There are two significant challenges as follows:

a) **Few-Shot Multi-Task problem**. FedAvg assumes that a single client already has a certain amount of training data, and the client's motivation is to reduce the generalization error of local data with the help of other clients. However, this assumption does not hold in many real cases. The data generated by each device is related to the environment of the device and the usage habits of users on the device, which leads to the limitation and one-sidedness of the data. Consider an image classification setting shown in Fig. 1: the mobile smartphone *A* needs to classify house and horse, while mobile smartphone *B* wants to classify bird and dog, and the tasks of other smartphones are also different. Each of these smartphones has only a few shots of pictures and wants to output multi-category prediction results. There are two major differences between this setting and that of FedAvg: (1) The number of training samples on a single device is too small to train a model, also as known as the Few-Shot Learning problem; (2) The training tasks of devices vary, as known as Multi-Task Learning problem. If the training samples of all devices are gathered together, the target size of the model will suffer from the curse of dimensionality.

b) **Credibility and Security issues**. Conventional distributed learning methods usually assume that the devices are under the centralized control of a trusted central server, and all clients will execute in strict accordance with the instructions of the server. However, in federated learning, the devices have absolute control over their data and behavior due to autonomy and intelligence. What they expect is to maximize their own interests instead of the global interests. Due to selfishness, the devices may not be able to elect a trusted third-party that everyone is satisfied with as the central server, or they may launch malicious attacks by fabricating model parameters to bias the model toward

a certain device. Therefore, it is reasonable and necessary to consider the mutual distrust between these devices. Lyu et al. [5] found that there are many security risks in the distributed scene of federated learning. For example, malicious attackers can use certain security vulnerabilities in federated learning architecture to damage the model effect. Besides that, as the central server is prone to a single point of failure or is not neutral (more inclined to some devices), that may greatly affect devices' willingness to participate [6].

Meta-learning is a natural choice to handle the few-shot multi-task problem. Since human beings have prior knowledge and the ability to rapidly adapt, they need only a few samples for each category of images to accurately classify the unknown samples. Based on this intuitive idea, some researchers put forward meta-learning [7, 8]. In meta-learning, prior knowledge is called meta-knowledge, and rapid adaptation is the goal of model optimization. The motivation of meta-learning is no longer to optimize the loss of an individual task but to learn how to learn. That is to say, meta-learning can quickly get a model with better performance for unknown tasks based on prior knowledge and a few shots of samples. Different from current Federated Meta-Learning works that generate multiple models to solve the personalization problem, we aim to train one global model that can be applied to multiple tasks. In this paper, we incorporate meta-learning into the federated learning framework, and conduct a federated meta-learning framework for the few-shot multi-task problem in a distributed setting, expecting to obtain an effectively shared global model which can rapidly adapt to different tasks.

The current federated learning frameworks are usually vulnerable to malicious attacks, from both clients and servers. The current works can only solve one of these two problems. The works that consider not only the trustworthiness of the server but also the security problem of the clients at the same time are very limited. Considering the credibility and security issues, we propose a decentralized federated Meta-Learning framework. In particular, considering that there is no trusted third party, we directly eliminate the central server in the conventional federated learning architecture and allocate the responsibilities of the central server to some elected committee nodes. Besides, considering the existence of malicious devices, the committee nodes conduct cross-validation and secure aggregation mechanism to avoid the security risks brought by them. Since communication overhead is a major bottleneck in federated learning, and the committee strategy will produce additional communication overhead, we provide a comprehensive analysis of the time complexity of the proposed framework, and propose a feasible scheme to reduce communication overhead.

From the perspective of the training framework, the comparison between DFMLF and other frameworks is listed in Table 1, where "-" means that there is no such problem, "✗" means that the framework can not solve the problem, and "✓" means that the framework can effectively solve the corresponding problem.

TABLE 1 The functional comparison between DFMLF and other frameworks.

Frameworks	data privacy	malicious device	no-trust server	few-shot multi-task
Pre-train [9]	-	-	-	✗
Centralized [9]	✗	-	-	✓
FedAvg [4]	✓	✗	✗	✗
Fed-Meta [10]	✓	✗	✗	✓
DFMLF	✓	✓	✓	✓

The contributions of this paper are summarized as follows:

- We incorporate meta-learning into the federated learning framework to solve the few-shot multi-task problem of mobile devices. Compared with the conventional meta-learning, our framework enables decentralized mobile

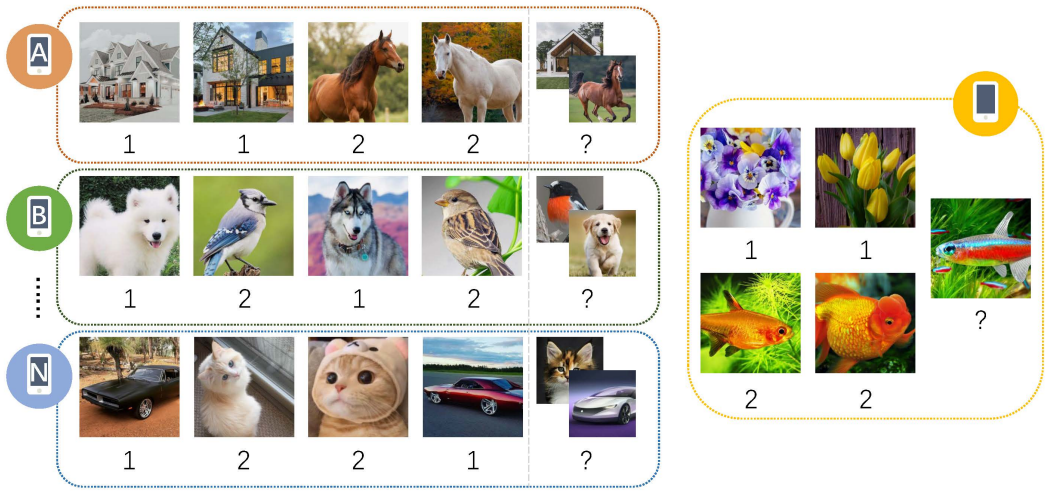

FIGURE 1 The distributed few-shot multi-task learning problem, each device has a few shots of data, and the training tasks of devices are different, the target dimension of the model is too tremendous to train.

devices to train a shared global model collaboratively without exposing their data to others, which can protect the privacy of data and is more suitable for real-world applications. Different from current Federated Multi-Task Learning and Federated Meta-Learning works, which learn separate personalization models for each client, our federated meta-learning framework expects to train one global model collaboratively to quickly adapt to different datasets (or tasks) on each device. In addition, our framework does not need to specify the model structure, so it is model agnostic, which makes it flexible under different settings.

- We consider not only the absence of a trust central server, but also the existence of malicious devices, and we propose a decentralized federated meta-learning framework, which directly allocates the responsibilities of the central server to some elected committee nodes. This server-less design can avoid the credible problems brought by the untrusted central server. Besides, the committee nodes can cross-verify whether the clients' models have the ability of rapid adaptation, and avoid the attack of malicious devices through filtering and secure aggregation, so that participants can spontaneously supervise and promote the training process.
- Communication cost is a key concern in federated learning. With the committee strategy, extra communication costs are expected. We discuss the time complexity of communication and computation for the proposed framework in detail. Then we propose a communication-efficient method based on the timing model, which relaxes the iteration dependency to 2, so that the training of training nodes and the aggregation of committee nodes can be carried out in parallel between adjacent iterations while maintaining global synchronization.
- To verify the effectiveness of DFMLF, experiments are carried out on a real-world dataset with different tasks and different basic models, and the model performances are demonstrated and analyzed. We also design some simulation experiments with malicious attacks, and verify the robustness of DFMLF by comparing it with other robust methods. Besides that, experiments show that our communication-efficient method greatly reduces the communication overhead, and our proposed framework is more efficient than Krum function.

The remaining of this paper is organized as follows. Section 2 surveys the related work. Section 3 introduces our

framework. Experimental results and analysis are summarized in Section 4. Finally, we conclude the paper in Section 5.

2 | RELATED WORK

2.1 | Meta-Learning

Meta-learning, which only needs a few shots of samples for each category of images to accurately classify the unknown samples [7, 8], can get a model with good performance for the few-shot multi-task problem. Meta-learning is a subfield of machine learning, which aims to learn meta-knowledge to help the process of modeling and training new tasks. Meta-knowledge can be expressed as the method of planning, modeling, labeling, learning, and updating, which usually exists in the model as a priori. Researchers use different methods to represent the abstract meta-knowledge into data that computers can understand, such as initialization model parameters or the model structure[11]. In deep learning, the objective functions are usually non-convex. This property makes it difficult for the optimization method based on the backpropagation gradient to find the global optimum in the parameter space, which usually falls into the local optimum or stagnates where the saddle point has a smaller gradient. To solve such non-convex optimization problems, the initial values of model parameters are crucial. Proper initialization model parameters can make the neural network model quickly converge to an optimal point. MAML (Model-Agnostic Meta-Learning)[7, 9, 12] defined what are good initial model parameters, that is, only a few gradient descent steps are needed to achieve good results on a new task. This is a solution to prevent training overfitting for the few-shot multi-task learning problem.

2.2 | Federated-Learning

Federated learning allows multiple devices distributed in different locations can build a shared global machine learning model while keeping their data locally. A conventional federated learning architecture generally includes a central server and multiple clients. The clients have absolute control over their data, while the central server is only responsible for parameter aggregation and exchange, and can not directly or indirectly access the data on the clients.

In recent years, security issues of federated learning have gradually attracted attention. Some malicious attackers make modifications at the data or model level to affect the performance of the global model. For example, Fung et al. [13] proposed the label flip attack, which uniformly flips the label of a certain category into another label, so that the model predicts the samples of this category with a high probability as another specified category. Gu et al. [14] proposed a more feasible solution, in reality, called Backdoor Poisoning, which only modifies a part of certain features of the training dataset, such as adding a small watermark to the picture. When processing normal samples, the model behaves normally, and when it encounters adversarial samples, the model will make incorrect predictions. This attack method is more difficult to detect and defend than label flip attacks. Bhagoji et al. [15] presented that the attacks against the model improve the concealment and anti-reconnaissance ability, and are more effective than the attacks against the data. Bagdasaryan et al. [16] proposed that malicious devices can perturb or modify the model parameters before uploading the gradient to achieve the goal of harming the global model. Moreover, Blanchard et al. [17] proposed a Byzantine attack, in which a series of malicious devices will collude to submit similar model updates, leading the global model in a wrong direction. Due to a large number of malicious devices, it is more difficult to defend than single-source attacks. Besides, Kang et al. [18] proposed that the workers may inadvertently generate low-quality updates that adversely affect the effectiveness of federated learning due to the high mobility or energy constraints.

Most of the aforementioned attacks are initiated by the training devices. However, studies have shown that the

central server may also initiate malicious attacks [5]. Some researchers have proposed decentralized federated learning. For example, Pappas et al. [19] built a fully decentralized federated learning framework based on the interplanetary file system (IPFS), in which each client retains a part of the model. Before training, each client will get other parts of the model from other clients, combine them into a complete model, and then send updates to other clients after training. Hu et al. [20] used the gossip protocol to ensure that the data of all nodes are the same, and to complete the upload and download processes in the conventional federated learning, so as to replace the central server. Roy et al. [21] proposed BrainTorrent, a server-less Peer-to-Peer federated learning environment, where all clients directly interact with each other. Li et al. [22] proposed a decentralized federated learning framework, which uses blockchain for the global model storage and the local model update exchange. However, these decentralized methods will inevitably bring huge communication overhead. Ng et al. [23] and Lim et al. [24] proposed Serverless Hierarchical Federated Learning (SHFL) framework, which adopts a two-layer system architecture. In the lower layer, the devices are grouped into clusters under cluster heads. In the upper layer, the cluster heads exchange the intermediate parameters with their one-hop neighbors without the aid of a central server. However, the hierarchical federated learning framework does not apply to the Few-Shot Multi-Task Learning problem, as the number of training samples on a single device is too small and the training tasks of the devices are different, the cluster method will reduce the effect of meta-learning.

Besides, huge communication and computation overhead is also a problem in federated learning. Many effective federated learning methods have been proposed to reduce the communication and computation costs. For example, Yan et al. [25] employed network pruning operation to accelerate the convergence of training; Hu et al. [26] proposed to extract the updated information of all the clients' models and train an auxiliary model on the server to realize information aggregation via the technique of Knowledge Distillation (KD), which can reduce the required computing resources of clients. Wang et al. [27] proposed a new method for parameters aggregation called orthogonal gradient aggregation, which removes the corresponding training samples while protecting the previously learned knowledge, so as to reduce the computation and communication overhead.

2.3 | Federated Multi-Task Learning

Some recent works have integrated federated and multi-task learning. Humbeck et al. [28] utilized the federated multi-task feedforward neural network to preserve the privacy of highly confidential and competitive data in the field of multi-task learning. Smith et al. [29] introduced a federated multi-task framework to address the statistical challenges in the federated setting. They see the personalization problem of federated learning as a multi-task learning problem, and the optimization on each client is considered as a new task. Mills et al. [30] proposed a multi-task learning approach to achieve personalisation in FL, which introduces non-federated Batch-Normalization layers into the federated Deep Neural Networks. Li et al. [31] focused on the impractical and inefficient problem of the federated multi-task algorithm in online scenarios (for example, when new mobile devices continue to join mobile computing systems), and proposed an online federated multi-task learning algorithm. Corinzia et al. [32] developed an algorithm for federated multi-task learning with non-convex models using approximated variational inference.

The goal of the above federated multi-task frameworks is to learn separated models for each client, which is significantly different from the conventional federated learning works [2, 3], which aim to train a single global model across the network. Besides, these federated multi-task frameworks rely on a central server to compute the relationships amongst tasks. While in practice, the existence of the central server is sometimes not feasible because of the competitive relationship of the clients. In order to eliminate the dependence on the central server, Dinh et al. [33] proposed a decentralized version of federated multi-tasking learning framework, and He et al. [34] introduced a Decentralized Multi-Task Learning Correlation Matrix Exchanging Algorithm. As there is no server for coordinating the learning, each

client is required to send its updated model to its neighboring clients in each round of training. Thus, the above frameworks are vulnerable to the client attacks.

2.4 | Federated Meta-Learning

Meta-learning is consistent with federated learning, and both are optimization problems of distributed datasets (or tasks). Jiang et al. have demonstrated theoretically that the two have a mathematical connection [35]. Chen et al. [10] and Lin et al. established the similarities between meta-learning and federated learning [36]. Some researchers have put forward federated meta-learning in distributed settings. Currently, most federated meta-learning focuses on the personalization model. For example, Alireza et al. [37] and Jiang et al. [35] proposed personalization federated learning via meta-learning. They regard the datasets on different devices as different tasks, and use the adaptability of meta-learning to generalize a single global model into multiple personalized models, so as to get models that adapt to different data distributions. Considering the slow convergence speed and low communication efficiency of federated meta-learning in the field of edge learning, Yue et al. [38] developed a non-uniform device selection scheme to accelerate the convergence. Then, they formulate a resource allocation problem to minimize the wall-clock time along with energy cost.

In addition, there are some works to consider malicious attacks in federated meta-learning. For example, Aramoon et al. [39] proposed a secure aggregation method in a federated meta-learning framework, which adds a layer of security aggregation to resist backdoor attacks. However, the architecture is still a classic federated architecture, which contains a central server. Bonawitz et al. [40] proposed a gradient encryption aggregation method from the perspective of encryption, but did not consider the existence of malicious devices. Fung et al. [13] proposed a method to modify the aggregation weight to reduce the influence of malicious devices. However, it requires the device to submit the gradient average of all historical updates, which will lead to a significant increase in storage and network transmission.

Different from Federated Multi-Task Learning and Federated Meta-Learning, this article focuses on Federated multi-task learning on a large number of mobile devices with only a few shots of samples. Most of Federated Multi-Task Learning and Federated Meta-Learning generate multiple models to solve personalization problem, while we aim to train one global model that can be applied to multiple tasks. In addition, we consider not only the trustworthiness of the server, but also the security problem of the clients. In this article, we consider a novel server-less federated meta-learning architecture that eliminates the central server to avoid the security risks caused by malicious central servers. At the same time, we design a dynamic committee election strategy, which elects a small number of committee nodes instead of the conventional central server to aggregate updates. Moreover, these committee nodes will evaluate all the updates, and only highly reliable updates are aggregated to prevent training devices from doing evil.

3 | OUR FRAMEWORK

In this section, we firstly describe the problem definition and notations. Then we introduce the overview of our framework. In sections 3.3 and 3.4, the specific training processes of training nodes and committee nodes are described in detail respectively, including the meta-training of the training nodes, committee verification, aggregation, and election of the committee nodes. In section 3.5, we analyze the time complexity of communication and computation, and propose a communication-efficient method for our framework.

3.1 | Problem Defination and Notations

We assume there exist N devices $\{C_1, C_2, ..., C_N\}$ which meet the following conditions:

- Any two devices are network reachable;
- Each device has local training tasks and local dataset;
- Each device has the ability of model training;
- Each device is reluctant to share its local dataset.

The tasks and local dataset of C_i are represented as $\{T_i\}$ and $\{D_i\}$ respectively. Each task has a few shots of data samples with labeled data, that is $D_i = (X_i, Y_i)$. We divide the dataset $\{D_i\}$ into Support Set D_i^{spt} and Query Set D_i^{qry}. D_i^{spt} is used for the training of the model, and D_i^{qry} is used to verify the model adaptation and update the meta-learner. In order to verify the ability of the model to quickly adapt to other tasks, we assume that there are some new nodes, called test nodes, that are not involved in the model training, while having local training tasks and need rapid adaptation. The number of test nodes is recorded as M, and the test nodes are recorded as $\{C_{N+1}, C_{N+2}, ..., C_{N+M}\}$, the symbols of the task sets and datasets on the test nodes also follow the same definition as the above training node. As shown in Fig. 2, this is an example of image classification. There are 12 training nodes $C_1, C_2, ..., C_{12}$, they have different classification tasks, and a few shots of images. For example, for training node C_9, task T_9 is to classify house and horse, D_9^{spt} provides a few shots of labeled images of house and horse to the model for training, and D_9^{qry} is used to verify the training adaptation; for training node C_{10}, task T_{10} classifies dog and bird, D_{10}^{spt} and D_{10}^{qry} provide images of dog and bird for training and verification respectively. According to the general definition of meta-learning, we call the number of classification targets of a task as n-way, and the number of training samples for each category is called k-shot. In the above example, the two-category task means 2-way, D_i^{spt} provides two samples for training, then the task is called a 2-shot task. For an n-way k-shot task, the dataset size is $|D^{spt}| + |D^{qry}| = 2nk$, where $|\cdot|$ refers to the number of elements in a set.

As each task has only a few shots of images, which is not enough to train a model, thus it is necessary to train the model collaboratively. The model is represented as $F_\theta(x)$, where F is the basic model that completes a specific task, θ is the parameter vector of the model F, and x is the input sample feature. The motivation is to find a globally shared model parameter vector θ, which can quickly adapt to different tasks. In our framework, there is only one restriction on the basic model: the objective function is differentiable. In view of this, most models based on gradient descent can be used, including the Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), and so on. We assume that the architecture of the model F and its hyperparameters (such as the learning rate) are consistent on each node. In addition, if all tasks are irrelevant, no meta-knowledge can be transferred. Therefore, we assume that the above tasks are subject to a potential distribution of $T_i \sim p(T)$.

3.2 | Framework Overview

The framework has two kinds of node roles: training nodes and committee nodes. Devices can change roles and complete different tasks. Because the devices can be connected to each other, they can form an alliance in which some protocols or contracts, such as Smart Contract and Practical Byzantine Fault Tolerance (PBFT) protocol [41], are applied to realize role management and switching.

- Training node: The training nodes use local datasets for training and are responsible for local updating of model

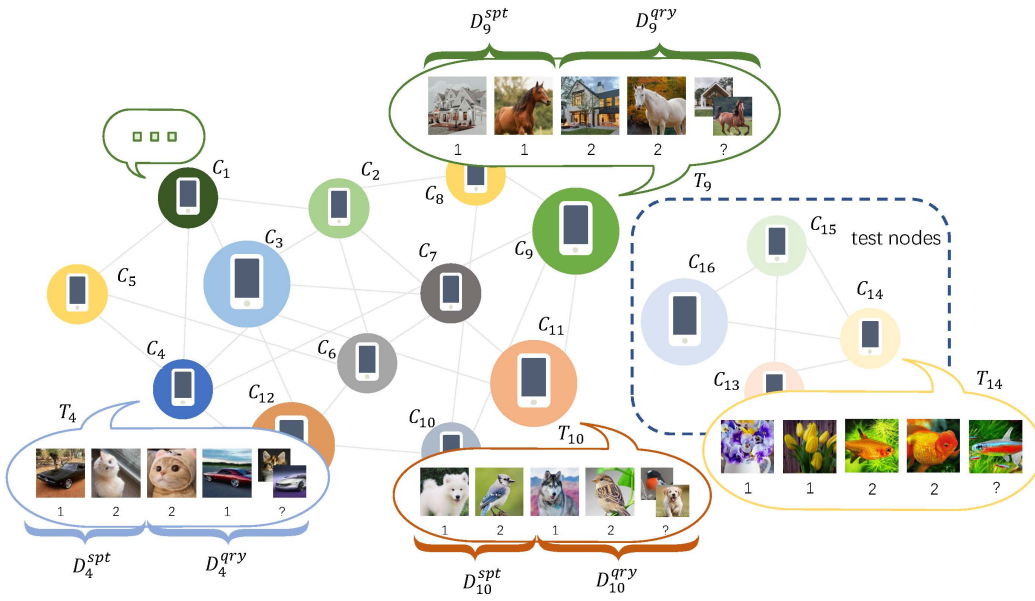

FIGURE 2 There are 12 training nodes and 4 test nodes. The task on each node is different.

parameters;

- Committee node: The committee nodes are responsible for the validation and aggregation of model parameters.

In each round, some devices play the role of committee node. In the first round of training, the devices willing to become committee nodes can be committee nodes. And in the next rounds, the committee nodes will be elected according to a certain selection strategy. In the meantime, some devices play the role of training node. These training nodes can be spontaneous or selected by the committee nodes.

We formally define the overview framework as Fig. 3, where t represents the t-th global round of federated learning:

(1) Each training node C_i receives the current global model parameter vector θ^t, and performs rapid adaptation and adaptation verification based on their local support set D_i^{spt} and support set D_i^{spt} respectively, and then updates the model parameter vector θ^t through the meta-learner to quickly adapt to the new tasks.

(2)After training, each training node C_i sends the updated model parameter vector θ_i^t to all the committee nodes.

(3) After a period of time, or the committee nodes receive a certain amount of the updated model parameter vector θ_i^t send by training nodes. The committee node needs to reach a consensus on filtering malicious training nodes and secure aggregation. We consider the situation of heterogeneous networks and heterogeneous devices, that is, the transmission speed and training time of training nodes are various, resulting in a different set of θ_i^t received by each committee node. Therefore, each committee node should transmit the model parameter vector it receives to other committee nodes, so as to make all committee nodes have the model parameter vector sent by the training nodes that can be received. Then, each committee node begins to validate the updated model parameter vectors it receives. After verification, each committee node would possess the scores of the model parameter vectors submitted by training

TABLE 2 Notations and explanations

Modules	Notations	Description
Node	N	the number of nodes
	M	the number of test nodes
	C_i	i-th node
Dataset	T_i	the task of node C_i
	D_i	the dataset of node C_i
	X_i	the input feature of node C_i
	Y_i	the output label of node C_i
	D_i^{spt}	the Support Set of node C_i
	D_i^{qry}	the Query Set of node C_i
Model	F	the architecture of the basic model
	θ	the parameters of the basic model
	$\mathcal{L}$	the loss function of the basic model
	n	the number of classification targets of a task
	k	the number of training samples for each category
	τ	the number of local gradient update steps
	α	the learning rate for rapid adaptation
	β	the learning rate for meta-learner
Committee	N_{train}	the number of training nodes
	N_{comm}	the number of committee nodes

nodes. These scores reflect the ability of the model parameter vectors to adapt to new tasks quickly. To filter out malicious attacks, the committee nodes sort the training nodes according to the verification scores in descending order, and select the model parameter vectors of the non-Byzantine training nodes to aggregate the new global model parameter vector.

(4) After global aggregation, the committee nodes need to be re-elected. We define that some non-Byzantine training nodes are randomly selected as the committee nodes of the next round according to the verification scores.

(5) The current committee nodes become common nodes and can participate in the next round of training.

(6) The training nodes are selected for the next round.

The whole program will continue until converging to an optimum.

3.3 | Training Nodes

We wish to train an initial global model in meta-learning, which can quickly adapt to other tasks. That is, the initial global model is expected to have a better generalization performance on an unseen task during the testing phase by updating with a few examples. Therefore, we should find model parameters sensitive to task changes, thus, small changes in parameter vector will greatly improve the loss function on any task. The devices that want to participate in the training have been listening to the signal of the network and waiting to receive the global model parameter vector θ. After the training nodes download the global model parameter vector θ, they use a meta-learning optimization method based on gradient descent to update the global model parameter vector by using their local datasets. Meta-learning focuses on finding one initial global model parameter vector θ which performs well on all tasks, that is, the initial global model parameter vector θ has rapid adaptability, and it only requires a small number of training rounds to achieve good performance on new tasks. We divide the meta-learning process of training nodes into the following three steps:

Rapid Adaptation. In order to get the level of the rapid adaptability of the initial global model parameter vector θ, it should perform once or multiple adaptions on a new task T_i. The training node i performs training based on its local

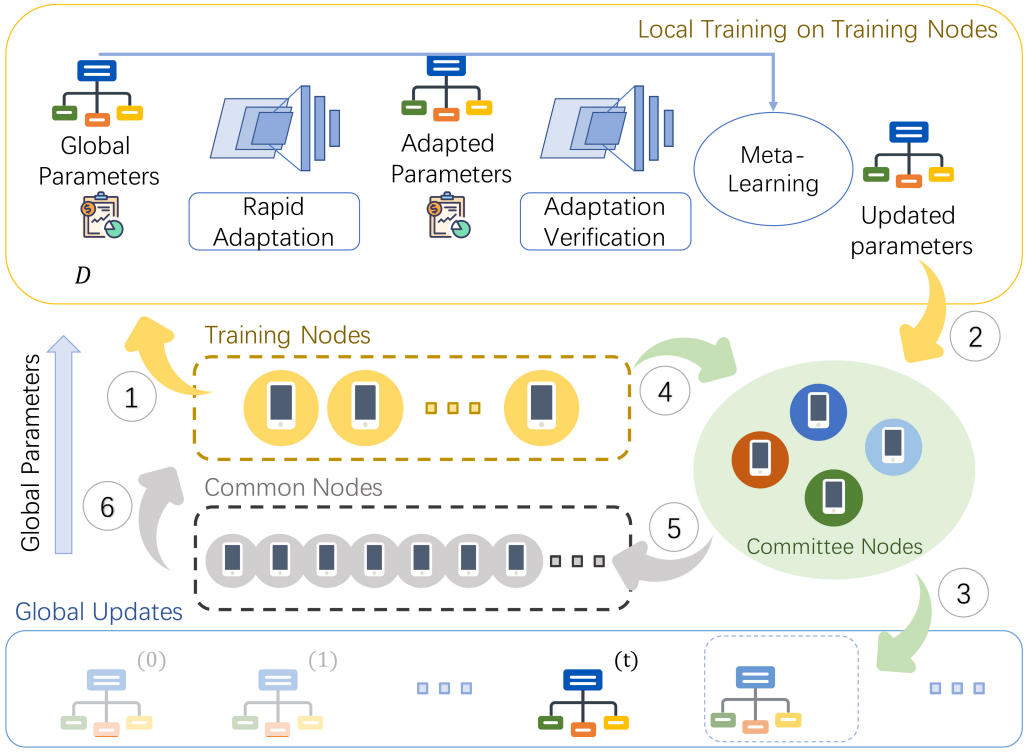

FIGURE 3 The Decentralized Federated Meta-Learning Framework for few-shot multi-task learning: (1) The training nodes obtain the latest global model parameter vector and conduct meta training locally; (2) The training nodes send the model updated parameter vector of local training to the committee nodes; (3) The committee nodes perform verification and aggregation; (4) Randomly select some non-Byzantine training nodes to become the committee nodes of the next round; (5) The current committee nodes become common nodes; (6) The training nodes are selected for the next round.

support set D_i^{spt} to update the initial global model parameter vector θ. We take the image classification as an example, the loss function is as follows:

$$\mathcal{L}(F_\theta, D_i^{spt}) = -\sum_{x_i} y_i \log \hat{y}_i, \tag{1}$$

where y_i is the ture label of input data x_i, and $\hat{y}_i = F_\theta(x_i)$ is the probability distribution of each category predicted by the basic model. Assuming that the loss function is differentiable, the training node i performs one gradient descent on the model parameter vector θ. The updating can be defined as:

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}(F_\theta, D_i^{spt}), \tag{2}$$

where α is the learning rate of rapid adaptation.

Adaptation Verification. After the updating of Eq. 2, the initial global model parameter vector θ has been adapted on the task T_i. We don't care about the performance of θ on task T_i, instead, we focus on the rapid adaptability of the updated global model parameter vector θ_i' which is updated based on θ. Therefore, the adapting effect of θ_i' is verified on the request set of the same task T_i:

$$\mathcal{L}(F_{\theta_i'}, D_i^{qry}) = \mathcal{L}(F_{\theta - \alpha \nabla_\theta \mathcal{L}(F_\theta, D_i^{spt})}, D_i^{qry}). \tag{3}$$

As shown in Eq. 3, the training variables are still the model parameter vector θ in the loss function of the verification phase. The physical meaning is the performance of θ after adapting to the new task.

Meta-Learner. Since our aim is to learn a model parameter vector θ, which can make rapid progress on new tasks, we should find model parameter vector sensitive to task changes. In this way, small changes in parameter vector will greatly improve the loss function on any task. The meta-learner optimizes the Eq. 3 based on gradient descent to achieve the goal of rapid adaptation. The optimization process of meta-learner can be written as follows:

$$\theta_i' \leftarrow \theta - \beta \nabla_\theta \mathcal{L}(F_{\theta_i'}, D_i^{qry}), \tag{4}$$

where β is the learning rate of the meta-learner. There is a derivative operation in the expansion of θ_i', thus the derivation of Eq. 4 will contain the term of second derivative. By further expansion, the gradient of the above formula can be rewritten as:

$$\begin{aligned}\nabla_\theta \mathcal{L}(F_{\theta_i'}, D_i^{qry}) =& \nabla_{\theta_\tau} \mathcal{L}(F_{\theta_\tau}, D_\tau^{qry}) \\ & \cdot \prod_{i=1}^{\tau} (I - \underbrace{\alpha \nabla_{\theta_{i-1}} (\nabla_\theta \mathcal{L}(F_{\theta_{i-1}}))}_{\text{second order}}),\end{aligned}$$

where τ is the number of local gradient update steps. In practical, the second order derivation can be removed to speed up the training of the model [7]. When the second order derivation is ignored, we get a first-order approximation of the update direction of the meta-learner:

$$\nabla_\theta \mathcal{L}(F_{\theta_i'}, D_i^{qry}) \approx \nabla_{\theta_\tau} \mathcal{L}(F_{\theta_\tau}, D_\tau^{qry}).$$

After training, the training node i sends the updated model parameter vector θ_i' to all committee nodes.

3.4 | Committee Nodes

In conventional federated learning, there is a central server to aggregate the updated model parameter vectors sent by training nodes. We replace the central server with some elected committee nodes. The framework eliminates the central server, thus getting rid of the possibility of the server doing evil. Meanwhile, the committee can effectively locate the malicious training nodes and invalidate them by sorting and filtering. Moreover, the dynamic committee election mechanism can further improve the robustness of model training.

In this section, We will introduce how the committee nodes filter the malicious training nodes; and how to reach a consensus among committee nodes to achieve secure aggregation; and then introduce the dynamic committee elec-

tion strategy.

3.4.1 | Validation

Blanchard et al. [42] proposed a Byzantine worker can force the central server to select the updated model parameter vector whose direction is too far away from others, so that the average based methods cannot converge. To tolerate Byzantine workers, many works have been proposed to choose appropriate updates sent by the workers. For example, Chen et al. [43] suggested using the geometric median of the received updated model parameter vectors instead of the average for aggregation. Blanchard et al. [42] proposed Krum function, which combines majority-based and distance-based methods to select the updated model parameter vectors closest to a certain number of neighbors to aggregate, which is usually the number of non-Byzantine workers.

Most of these works assume that the data owned by participants are i.i.d, which is not in line with the actual situation of federated learning. Because in federated learning, the amount of participants' data is very small compared with the whole data, and the data distribution is different from the whole data. Therefore, the local model will deviate from the global model, and there will be great differences between local models. Different from the above works, in our framework, the committee nodes have their own data and can use their own data to judge the quality of the model, so as to more accurately identify malicious nodes and even rank the degree of these malicious nodes.

As mentioned in the previous section, each node only has a small amount of training data, and the optimization goal of the meta-learning is to quickly adapt to new tasks. The physical meaning of one good global model parameter vector θ is that it is a better initial parameter vector that can rapidly adapt to new tasks. In this section, we design a verification method for rapid adaptability of the global model parameter vector θ. Assume C_j is a committee node, and it evaluates the rapid adaptability of the model parameter vector θ.

First, the committee node C_j uses θ to adapt to the its task based on the local support dataset:

$$\theta' = \theta - \alpha \nabla_\theta \mathcal{L}(f_\theta, D_j^{spt}), \tag{5}$$

where θ' is the updated model parameter vector fitted on committee nodes. Then θ' is used to complete the forward propagation and obtain the prediction results on the query dataset D_j^{qry}:

$$\hat{Y} = F_{\theta'}(\hat{X}(D_j^{qry})), \tag{6}$$

where $\hat{X}(\cdot)$ is the operation of extracting training sample features. The predicted value of $\hat{y}$ will be compared with the real value of y:

$$Accuracy = \frac{1}{|Y|} \sum_{i=1}^{|Y|} \mathbb{I}(\hat{y}_i = y_i), \tag{7}$$

where $\mathbb{I}(\cdot)$is the indicator function, which returns 1 when $\hat{y}_i = y_i$, and 0 otherwise.

3.4.2 | Secure Aggregation

After verification, each committee node gets the scores of all the model parameter vectors submitted by training nodes. These scores reflect the ability of the model parameter vectors to adapt to new tasks. However, the order of the score set owned by each committee node is different. This is largely due to two reasons: 1) the datasets and task sets of committee nodes are different, the adaption ability of the same model parameter vector on different new tasks are varied; 2) some committee nodes may be malicious or attacked by malicious nodes, they give high scores to malicious training nodes, or give low scores to honest training nodes. Therefore, we need to design a robust strategy to accurately filter malicious training nodes in the case of Byzantine committee nodes, so as to achieve secure aggregation.

We have a basic intuition, that is, in all the score sets of non-Byzantine committee nodes, the scores of all malicious training nodes are lower than that of all honest training nodes. We assume that N_{train} is the number of training nodes, f is the number of Byzantine training nodes, N_{comm} is the number of committee nodes, $\overline{f}$ is the number of Byzantine committee nodes.

The overview of our secure aggregation algorithm is as follows:

Step 1: A committee node is selected as the main committee node by rotation or random algorithm.

Step 2: The main committee selects the training node with the highest score in the current period, and creates a request $< Request, training\text{-}node\text{-}id, operation, timestamp >$ to ask whether its model can be aggregated. Then the main committee node broadcasts the request to all other committee nodes. The operation in the request is to aggregate the updated parameter vector of the selected training node to the global model parameter vector as Eq. 8:

$$\theta^{t+1} = \frac{1}{N_{train} - f}\theta_i^t, \tag{8}$$

where θ_i^t represents the model parameter vector submitted by training node C_i in the t-th round. Note that Eq. 8 is different from the the federated aggregation in FedAvg [44], Eq. 8 does not consider the weight of the node's dataset size. The reason is as follows: (1) for security, malicious attacks can control their weights by falsely reporting the size of the dataset; (2) for privacy, the dataset size is one of the data attributes, which is privacy sensitive and may reduce the difficulty of reverse engineering of malicious attacks; (3) for small sample settings, the data sizes of nodes are similar, and each node only has a small amount of data.

Step 3: All committee nodes except the main committee node process the request. Each of them checks whether the selected training node is in the top $N_{train} - f$ nodes with the highest score. If so, it performs aggregation operation as Eq. 8 and checks whether the result is consistent with the request. If so, the processing result

$< reply, timestamp, training\text{-}node\text{-}id, response >$ is returned to the main committee node.

Step 4: The main committee node checks whether it has received at least $\overline{f} + 1$ identical results from other committee nodes. If it receives $\overline{f} + 1$ messages and the consensus is reached in these messages, we can consider that the consensus is reached in the current period.

Step 5: If the request is successful, this indicates that the aggregation operation of the selected training node has been completed on all committee nodes; otherwise, if the original request is denied, this illustrates that the main committee node may be a malicious node, so it is necessary to reassign the main committee node.

Step 6: The process iterates until the model parameter vector with $N_{train} - f$ training nodes are aggregated. After the global aggregation, θ^{t+1} will be sent to the training nodes as the starting point of the next round of training.

We analyze the security of our aggregation algorithm from two perspectives:

- Security of committee nodes: because there are $\overline{f}$ Byzantine committee nodes, according to the principle that the

minority obeys the majority, the number of non-Byzantine committee nodes $N_{comm} - \overline{f}$ only needs one more node than $\overline{f}$, that is, $N_{comm} - \overline{f} = \overline{f} + 1$. The number of non-Byzantine committee nodes will be more than the number of Byzantine committee nodes, so the committee can reach a consensus. The total number of committee nodes is $N_{comm} = \overline{f} + (\overline{f} + 1)$, the maximum number of Byzantinet-tolerant committee nodes supported in this case is $(N_{comm} - 1)/2$.

- Security of training nodes: as mentioned above, for non-Byzantine committee nodes, the scores of malicious training nodes are lower than that of non-Byzantine training nodes. That is, if we sort the training nodes according to the verification scores in descending order, for any non-Byzantine committee node i, the list of training nodes it owns is as follows:
  - $[1, 2, 3, ..., N_{train} - f]$ be indexes of non-Byzantine training nodes;
  - $[n - f + 1, ..., N_{train}]$ be indexes of Byzantine training nodes.

  We can see that for any non-Byzantine training node $j, j \in \{1, 2, 3, ..., N_{train} - f\}$, regardless of the number of Byzantine training nodes f, j is always in the top $N_{train} - f$ of the list of all the non-Byzantine committee nodes. Therefore, as long as the number of Byzantine committee nodes $\overline{f}$ is lower than $(N_{comm} - 1)/2$, the non-Byzantine committee nodes can filter out any number of malicious training nodes successfully.

3.4.3 | Dynamic Committee Election Strategy

The local tasks of the committee nodes can not represent the distribution of global tasks, so the static committee is not conducive to improving the generalization of the model. We design a dynamic committee election strategy: in each global round, after global aggregation, N_{comm} training nodes will be randomly selected from the non-Byzantine training nodes as the committee nodes of the next global round. Then, the current committee nodes become training nodes and can participate in the training tasks of the next global round. Here, we assume that the number of committee nodes N_{comm} is less than the number of non-Byzantine training nodes $N_{train} - f$, as a small committee will reduce the amount of calculation and communication in the phase of validation and aggregation.

The advantages of our dynamic committee election strategy are as follows:

Robustness and fairness : the validation dataset is replaced after the general election of the committee nodes, which is similar to K-Fold cross-validation in classical machine learning training, and can improve the performance of model training when the total amount of data is small. In addition, it is noticed that we randomly select the Committee nodes from the training nodes instead of the training nodes with the highest scores as the Committee nodes. This is because the random selection strategy will give all honest nodes the opportunity to be selected as committee nodes, which improves the generalization and fairness of the model. The optimal strategy will lead the system to choose the nodes consistent with the data distribution of the initial committee nodes as the next round of committee nodes, resulting in poor generalization performance and unfairness of the final model.

Security : if the number of non-Byzantine committee nodes is greater than $N_{comm}/2$, we rank the training nodes according to the scores of the committee nodes on the training nodes, and the malicious training nodes are ranked after the honest training nodes. We only select committee nodes from honest training nodes, and malicious training nodes will not be aggregated or become the next committee nodes. In another case, a malicious training node may disguise itself as a normal training node before becoming a committee node, and it starts to attack the system after being selected as a committee node. Assuming that the probability that each training node is selected as a committee node is the same, in order to avoid the number of malicious committee nodes accounting for more than half, we need to ensure that the number of malicious training nodes is less than half. That is, $f < N_{train}/2$.

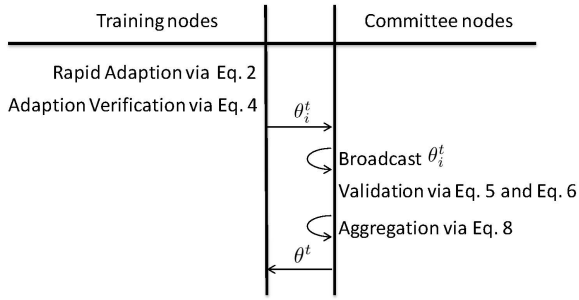

FIGURE 4 The t-th round of our algorithm

3.5 | Communicaiton-Efficient Method

In this section, we will analyze the complexity of our algorithm. The t-th round of the algorithm is summarized in Fig. 4, and described as follows:

(1) Each training node C_i receives the same global model parameter vector θ^t, and then performs rapid adaption via Eq. 2 and adaption verification via Eq. 4. After that, it sends the local updated model parameter vector θ_i^t to all the committee nodes.

(2) As the heterogeneous networks and devices, each committee node C_j only receives the local updated model parameter vectors from some training nodes. Each committee node C_j broadcasts the local updated model parameter vectors it receives to other committee nodes. Then, each committee node C_j begins to validate these local updated model parameter vectors via Eq. 5 and Eq. 6.

(3) After verification, the committee nodes sort the training nodes according to the scores in descending order, and select the local updated model parameter vectors of the non-Byzantine training nodes to aggregate the global model parameter vector together. Then, the new global model parameter vector θ^{t+1} will be broadcasted to all training nodes.

We analyze the time complexity of our algorithm in two aspects: communication and computation, T_{com} and T_{cmp} refer to the time taken by communication and computation, respectively.

3.5.1 | Computation Complexity

For training nodes : The main computation of training node is to evaluate the gradients of rapid adaption and meta-learner. The computational complexity of evaluating the gradients is $O(|D|\,|\theta|)$, where $|D|$ is the number of data samples, $|\theta|$ it the number of model parameter vector θ. Therefore, for each training node C_i, the computational complexity $T_{cmp-train} = T_{adaption} + T_{meta} = O(\left|D_i^{qry} + D_i^{spt}\right| * |\theta|)$.

For committee nodes : In the phase of broadcasting, the committee node's main workload is to compare whether the model parameter vectors it receives are consistent with that sent by other committee nodes, so the computational complexity of broadcasting $T_{broadcasting} = O(N_{comm} * N_{train} * |\theta|)$.

In the phase of validation, the committee node's main calculation is to evaluate the updated model parameter vectors sent by training nodes via Eq. 5 and forward propagation via Eq. 6. So the computational complexity of validation $T_{validation} = O(N_{train} * \left|D_j^{spt}\right| * |\theta| + |\theta|)$.

In the phase of aggregation, the main calculation of committee node is to compare whether the model parameter vectors are consistent and aggregate the model. Therefore, the computational complexity of aggregation $T_{aggregation} =$

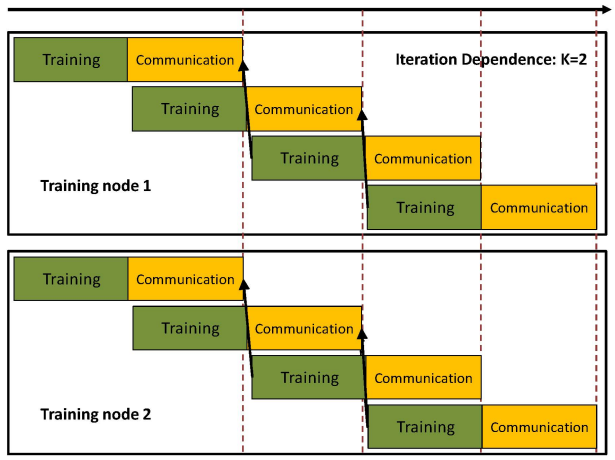

FIGURE 5 The Timing Model which relaxes the iteration dependency to 2

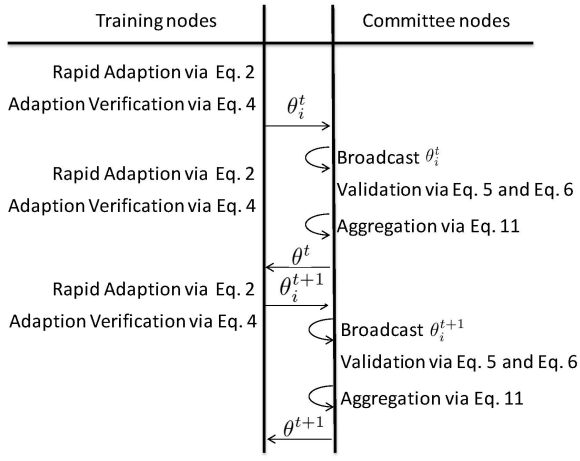

FIGURE 6 The communication-efficient method for our framework

$O(N_{comm} * N_{train} * |\theta|) + O(|\theta| * N_{train})$.

The computation time of Krum [42] is $O({N_{train}}^2 * |\theta| + N_{train})$. As the number of committee nodes N_{comm} is much less than the number of training nodes N_{train}, our aggregation algorithm is more efficient than Krum.

3.5.2 | Communication Complexity

The communication time is equal to the amount of data transferred s divided by the transmission rate r: $T_{com} = s/r$. As we can see from Fig. 4, four transmissions are required in one iteration.

First, each training node C_i upload its local updated model parameter vector θ_i^t to all the committee nodes, the uplink time $T_{com-up1} = O(N_{comm} * |\theta| / r)$; and the downlink time of each committee node is $T_{com-down1} = O(N_{train} * |\theta| / r)$.

Second, each committee node will send the local updated model parameter vectors it receives to all the other committee nodes, thus the uplink time $T_{com-up2} = O(N_{comm} * N_{train} * |\theta| / r)$; and the downlink time of each committee node is as same as the uplink time.

Third, the main committee node will send the honest model parameter vectors to all the other committee nodes, and the other committee nodes send back the aggregated global model parameter vector to the main committee node. Therefore, the uplink time $T_{com-up3} = O((N_{train} - f) * N_{comm} * |\theta| / r)$; and the downlink time is as same as the uplink time.

Fourth, the new global model parameter vector θ^{t+1} is broadcasted to all nodes. Therefore, the uplink time $T_{com-up4} = O(N * |\theta| / r)$; and the downlink time is as same as the uplink time.

3.5.3 | Communication-Efficient Method

We divide the total time required by the whole process into two parts, one is the training time of the training nodes, the other is the computation and communication time of the committee nodes. Then, for training nodes:

$$T_{train} = O(\left|D_i^{qry} + D_i^{spt}\right| * |\theta|); \tag{9}$$

for committee nodes:

$$T_{comm} = O(N_{train} * \left|D_j^{spt}\right| * |\theta| + N_{comm} * N_{train} * |\theta|). \tag{10}$$

Because the values of N_{comm}, N_{train}, $\left|D_j^{spt}\right|$, $\left|D_i^{qry}\right|$, and $\left|D_i^{spt}\right|$ are relatively small, we can see from Eq. 9 and Eq. 10 that the time required for the two parts is not much different. In conventional synchronous federated learning, after a round of training, the training nodes need to wait for the committee to aggregate and generate a new global model before they can start the next round of training. Thus, the total runtime is: $T = T_{train} + T_{comm}$. To address the long execution time for synchronous training, we use the timing model proposed by Li et al. [45], which relaxes the iteration dependency to 2, that is, each update depends on the updates of the 2-th last iteration. This makes the training of training nodes and the aggregation of committee nodes can be carried out in parallel and interleaved between neighboring iterations while maintaining global synchronous communication, as shown in Fig. 5.

However, in the timing model[45], all clients are required to participate in the training every round. Different from that, in our framework, the clients participating in training are different in each iteration. In order to achieve the consistency of training iteration, that is, the results of each training iteration can provide information for the next training iteration, we rewrite Eq. 8 as follows:

$$\theta^{t+1} = \frac{\theta^t + \theta_i^t}{2 * (N_{train} - f)}. \tag{11}$$

The communication-efficient method is shown as Fig. 6. The total runtime is: $T = max\{T_{train}, T_{comm}\}$, where the total runtime is solely determined by either training nodes or committee nodes.

Algorithm 1 illustrates DFMLF.

Algorithm 1: Decentralized Federated Meta-Learning Framework for few-shot multi-task learning

Input: N_{train}, N_{comm}, N, θ, learning rate
Initialize $\nabla\theta$ of iteration $[-1, 0]$ as zero and mark them as ready;
for t = 1,2,3,...,T **do**
  foreach $C_i \in \{C_{N_{train}}\}$ **do in parallel**
    Wait until the global model parameter vector $\nabla\theta^{t-2}$ is ready;
    Update $\theta_i^t = \theta^{t-1} - \gamma * \nabla\theta^{t-2}$;
    Perform rapid adaption via Eq. 2;
    Perform adaption verification via Eq. 4;
    Send the updated model parameter vector $\theta_{i'}^t$ to all the committee nodes;
  end
  foreach $C_j \in \{C_{N_{comm}}\}$ **do in parallel**
    Wait until some updated local model parameter vector $\theta_{i'}^t$ are ready;
    Broadcasts the received $\theta_{i'}^t$ to other committee nodes;
    Evaluate the $\theta_{i'}^t$ of training nodes via Eq. 7;
    Aggregate the new global model parameter vector θ^{t+1} via Eq. 11;
    Select some honest training nodes randomly as the committee nodes of the next round;
    The current committee nodes become normal nodes;
  end
end

4 | EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we present the experimental setup, including the real-world datasets, models, and environment in Section 4.1, and introduce compared methods in Section 4.2. Then we report experimental results and parameter analysis in Section 4.3 and Section 4.4, respectively.

4.1 | Experimental Setup

Considering system heterogeneity [46] and enormous participated devices [38] in federated learning, the number of clients participating in each round of training is small. In all of our experiments, we sample 1000 nodes to participate in training from the training set, the number of training nodes participating in each round is 1% of all training nodes, and randomly generate 10 test nodes from the test set to test the current model. These test nodes represent the node newly added, their local datasets have not been trained and can be used to evaluate the training performance of the global model. This ensures that the model evaluation is performed on unseen samples, so it can effectively reflect the generalization performance of the model.

4.1.1 | Datasets

We employ Omniglot and miniImageNet, two datasets commonly used in the field of few-shot learning research.

(1) Omniglot: Omniglot contains a series of handwritten character images, including 1623 different characters from 50 different languages. And each character contains 20 data sample images written by different people. Thus,

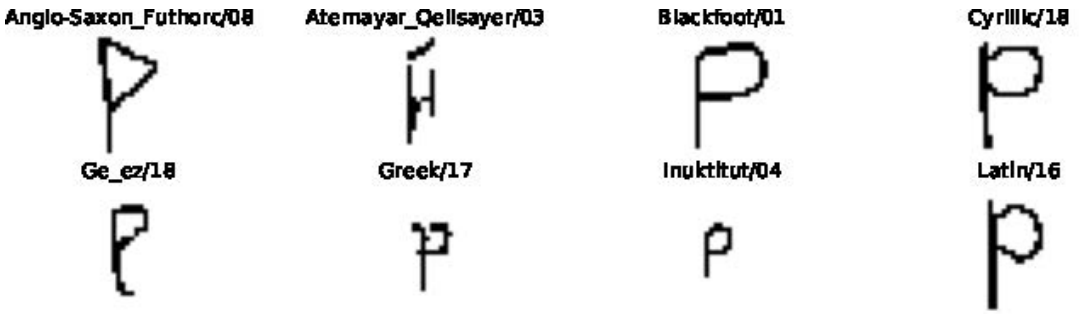

FIGURE 7 Similar characters written in eight different languages.

there are 32460 images in the whole dataset. In these images, some characters written in different languages may be similar. For example, Fig. 7 shows similar characters in eight different languages, all of which are in the form of English character p or Greek character ρ. In order to classify these images, we can train a recognition model for each language, or we can gather all the samples together to train a single model. However, the problem of the former is that the number of individual language samples is too small to train a model; while the latter problem is that the target dimension of the model will increase sharply, becoming a 1623 classification problem that is difficult to train, and classifying similar samples into other languages.

Faced with such a problem setting, we construct Omniglot into multiple different tasks and learn the common knowledge among them to complete the training of different tasks. For example, as shown in Fig. 8, we construct a 5-way, 1-shot task for each node. First, we randomly select 5 characters from the above 1623 characters. Then from the 20 samples of each character, we select 1 samples as the support set D^{spt}, and the other 1 samples as the request set D^{qry}. We divide 1623 characters into training set and test set, the sizes are 1423 and 200 respectively. The sampling of the training process will be performed on the training set, and the sampling of the new node will be performed on the test set.

(2) miniImageNet: miniImageNet contains 100 classes, each class has 600 examples, that is, a total of 60000 colour images of size 84×84. And we used 80 classes for training and tested on the remaining 20 classes. The experiments on miniImageNet revolve around the same basic task as Omniglot. That is, each client is assigned a set of $2 \times k$ examples from each of the n classes, k of which are the support set and the other k samples are the request set. Then, the task is to classify a disjoint batch of unlabelled examples into one of these n classes. The larger the number of n-way, the larger the dimension of classification task and the greater the difficulty of training; the larger the number of k-shots, the more training data, thus the less difficulty of training.

Most of the current centralized Meta-Learning methods [7, 47] generally use 1 or 5 shots to realize fast learning of n-way classification. For a 5-way, 5-shot task, each node should have 50 samples, which is often unrealistic in federated learning, as each client only has a very small amount of data. Therefore, we construct 1 or 2 shots for n-way classification, which is much more difficult than the centralized Meta-Learning methods.

As for n-way classification, the centralized Meta-Learning methods [7, 47] construct a 20-way classification for Omniglot, which is difficult to converge in federated scenarios, as there are only 10 nodes participating in training in each round in the Federated learning, which is far less than 20. Chen et al. [48] construct 5-way classification task for both Omniglot and miniImageNet in Federated Meta-Learning. In view of this, we construct "5-way, 1-shot", "5-way, 2-shot", "10-way, 1-shot", and "10-way, 2-shot" scenarios on the Omniglot, and "5-way, 1-shot", "5-way, 2-shot" scenarios on miniImageNet.

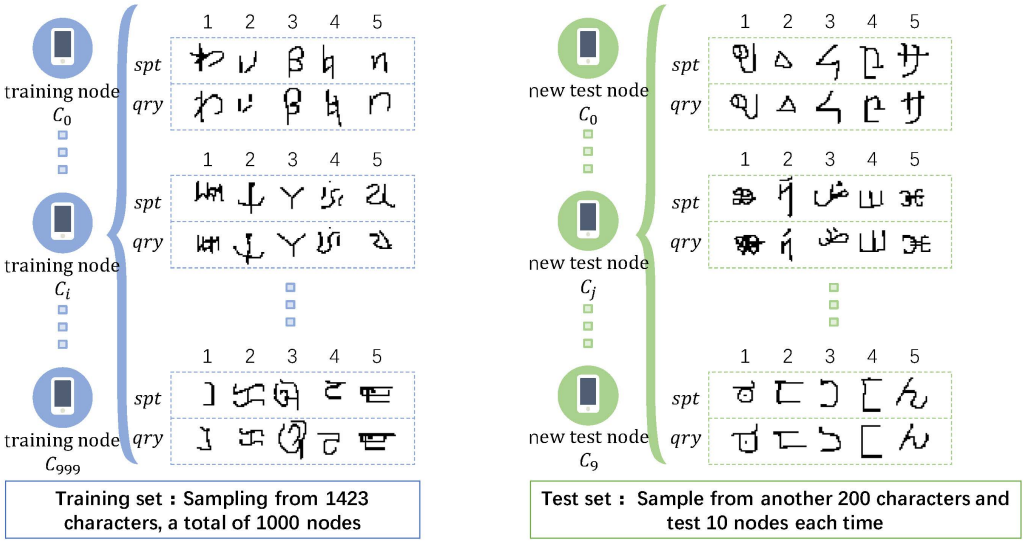

FIGURE 8 An example of 5-way, 1-shot task setting. In each task, the first line is the support set, and the second line is the query set, so there are a total of 2nk = 10 samples.

4.1.2 | Models

For Omniglot, we use two different basic models for training, CNN and MLP. The structure of the basic model CNN and MLP is shown in Table 3. In the model CNN, the four-dimensional tensor represents the width, height, channel number and convolution kernel number of the convolution layer, and the one-dimensional vector represents the offset, the total number of parameters is 111749; In the model MLP, the two-dimensional matrix of the full connection layer represents the input and output dimensions, and the one-dimensional vector represents the offset, the total number of parameters is 246597. The number of parameters of the MLP model is much more than that of the CNN model, so the difficulty of training also increases.

For miniImageNet, we use ResNet18 [49], which is a convolutional neural network that is 18 layers deep, and LSTM (Long short-term memory), which is a two layer LSTM binary classifier containing 64 hidden units.

We set the learning rate for rapid adaptation $\alpha = 0.1$, the learning rate for meta-learner $\beta = 0.001$ for Omniglot; and $\alpha = 0.01$, $\beta = 0.001$ for miniImagenet. We set the global iteration to 4000 rounds, the number of training nodes participating in each round $N_{train} = 10$, and the number of committee nodes N_{comm} is 40% of the number of training nodes participating in training. And in each round, we randomly generate 10 new test nodes to test the global model parameter vector and calculate the accuracy via Eq. 7, and then we use the average value of these 10 nodes as the test result.

4.2 | Compared Methods

We carried out experiments from three aspects: effectiveness, robustness and efficiency. In this section, we will introduce the comparison methods of these three aspects respectively.

TABLE 3 The structure of the base model CNN and MLP

#layers	CNN	MLP
1	(3, 3, 1, 64) + (64,)	(784, 256) + (256,)
2	(3, 3, 64, 64) + (64,)	(256, 128) + (128,)
3	(3, 3, 64, 64) + (64,)	(128, 64) + (64,)
4	(3, 3, 64, 64) + (64,)	(64, 64) + (64,)
5	(64, 5) + (5,)	(64, 5) + (5,)
	Total:111749	Total:246597

4.2.1 | Compared Methods of Effectiveness

For effectiveness, we will compare with several different training frameworks to verify whether our framework can effectively train the model in the distributed few-shot multi-task setting. The comparison training frameworks are:

- Centralized [7]: This framework collects all training tasks as a single dataset for training, and does not consider restrictions such as data privacy. This framework represents the upper bound of the model effect in federated learning [50, 51];
- Pre-train [7]: This framework is represented as the pre-training mode of a single model on all tasks. It regards all tasks as the same task, regards the data on different nodes as different batches of data, and trains the model serially on each node;
- Fed-Meta [10]: This framework is a classic federated learning architecture, including a central server and multiple training nodes. Each training node performs meta-learning, and uploads updates to the central server for averaging without verification process;
- DFMLF-Optimal: DFMLF is our proposed committee-based server-less decentralized federated meta-learning framework. In order to verify our random-based committee election strategy, we select the training nodes with the highest scores as the committee nodes in this comparison method.
- DFMLF-Random: In order to verify our communication-efficient method, we designed this comparison method for ablation experiment. This framework randomly selects committee nodes from honest training nodes without the communication-efficient method.
- DFMLF-Efficient: Our proposed framework with the communication-efficient method, which randomly selects the committee nodes from the honest training nodes.

4.2.2 | Compared Methods of Robustness

By simulating malicious attacks and comparing with other robust methods, we verify whether our framework can effectively resist malicious attacks against model parameter vectors and maintain the robustness of model training performance. These robust methods focus on the model aggregation stage to resist malicious attacks, and their training frameworks are unified as federated meta training [10].

- Naive: This aggregation method averages the parameters and does not consider the problem of malicious attacks.

TABLE 4 The performance of different frameworks based on the Omniglot dataset

Methods		5-way, 1-shot	5-way, 2-shot	10-way, 1-shot	10-way, 2-shot
CNN	Pre-train	28.71 ±1.03	28.20 ±1.08	14.62 ±0.62	15.37 ±0.58
	Centralized	95.49 ±0.41	97.12 ±0.30	87.34 ±0.45	86.18 ±0.60
	Fed-Meta	91.22 ±0.96	94.11 ±0.80	78.67 ±0.81	88.29 ±0.44
	DFMLF-Optimal	91.32 ±1.36	94.73 ±0.49	80.19 ±1.02	87.84 ±0.44
	DFMLF-Random	91.40 ±0.68	94.50 ±0.59	80.70 ±1.02	88.49 ±0.80
	DFMLF-Efficent	91.87 ±0.95	94.95 ±0.48	80.35 ±0.99	88.51 ±0.66
MLP	Pre-train	23.97 ±0.82	22.22 ±1.25	12.16 ±1.07	10.86 ±0.38
	Centralized	81.73 ±1.42	90.83 ±0.85	79.22 ±0.68	87.72 ±0.39
	Fed-Meta	85.03 ±1.51	92.31 ±0.59	84.15 ±0.78	84.07 ±1.39
	DFMLF-Optimal	84.59 ±1.05	90.91 ±0.95	82.64 ±0.92	87.89 ±0.40
	DFMLF-Random	85.48 ±1.12	93.46 ±0.52	84.51 ±0.72	89.15 ±0.54
	DFMLF-Efficent	86.68 ±1.07	92.90 ±0.56	84.90 ±0.74	88.52 ±0.52

TABLE 5 The performance of different frameworks based on the miniImagenet dataset

Methods	ResNet18		LSTM	
	5-way, 1-shot	5-way, 2-shot	5-way, 1-shot	5-way, 2-shot
Pre-train	22.25 ±1.04	22.46 ± 0.73	22.41 ± 0.80	22.53 ± 0.79
Centralized	40.18 ± 1.35	42.95 ± 1.48	33.56 ± 1.66	37.29 ± 1.36
Fed-Meta	32.28 ± 1.18	36.39 ± 0.96	31.22±1.30	34.23 ± 0.90
DFMLF-Optimal	34.08 ±1.56	40.32 ± 0.90	33.08± 1.22	31.89 ± 1.30
DFMLF-Random	34.06 ±1.42	39.58 ± 1.25	32.96± 1.31	34.62 ± 0.84
DFMLF-Efficent	34.17 ±1.36	39.31 ± 1.13	33.08±1.38	34.75 ± 0.81

- Median [43]: This aggregation method first sorts the parameters according to their values, and discards the maximum values of λ% and the minimum values of λ%, and then averages the remaining parameters, where $\lambda \in [0, 50)$;
- Krum [42]: This aggregation method assumes that the directions of the model parameter vectors submitted by honest nodes are relatively similar. By calculating the angle between the model parameter vectors, Krum selects the vectors closest to a certain number of neighbors to aggregate;
- DFMLF-Efficient: The proposed framework with the communication-efficient method, in which the committee nodes verify the parameter vectors of training nodes, and aggregate the local updates of the honest training nodes. And the committee nodes are randomly selected from the honest training nodes.

4.2.3 | Compared Methods of the Efficiency

In the DFMLF, the validation and secure aggregation of the committee nodes will incur additional communication and computation overhead. We propose a communication-efficient method to improve the wall-clock time. In order to clarify the efficiency of our method, we compare the following methods:

- Fed-Meta: This method is based on the classic federated learning architecture, which requires a central server, and does not consider the problem of malicious attacks;

- Fed-Krum: This method is also based on a centralized federated learning architecture. The central server needs additional calculations to filter malicious training nodes;
- BrainTorrent[21]: A new federated learning architecture without a central server, where all nodes directly interact with each other to share parameter vectors;
- DFMLF: The proposed framework without communication efficient method, and the responsibility of the central server is allocated to some committee nodes, such as filtering malicious training nodes and aggregation;
- DFMLF-Efficient: The proposed framework with communication efficient method, in which the training of training nodes and the aggregation of committee nodes are carried out in parallel.

4.3 | Experimental Results and Analysis

4.3.1 | Effectiveness

In this experiment, we do not consider the malicious attacks, and assume that every node is an honest node. In order to verify the performance in different situations, we constructed four different task settings for Omniglot: 5-way 1-shot, 5-way 2-shot, 10-way 1-shot, 10-way 2-shot; and two different task settings for miniImageNet: 5-way 1-shot, 5-way 2-shot.

From Table 4 and 5, the following observations and conclusions can be obtained:

The performance of the Pre-train is the worst, the prediction accuracy of the n-way task is only slightly higher than the random prediction $1/n$. This is because Pre-train does not consider that the tasks on different nodes are different, thus it cannot learn the common meta-knowledge of different tasks.

The Centralized mode ignores the restrictions of data privacy protection and gathers all tasks' data together. In conventional federated learning, the Centralized method is often be regarded as the theoretical upper bound. The accuracies of the Fed-Meta method in tasks 5-way 1-shot, 5-way 2-shot, 10-way 1-shot based on the CNN model, the task 10-way 2-shot based on the MLP model, and the tasks 5-way 1-shot, 5-way 2-shot based on the ResNet18 and LSTM model are about $3\% \sim 8\%$ lower than that of the corresponding Centralized method. While the accuracy of the Fed-Meta method on the task 10-way 2-shot based on CNN model, and the tasks 5-way 1-shot, 5-way 2-shot, 10-way 1-shot based on MLP model is about $2\% \sim 5\%$ higher than that of the corresponding Centralized method. This shows that although in theory, the federated method will lose global information compared with the Centralized method, the performance of the Fed-Meta method is better than the Centralized method in some tasks. That is because the model obtained by Fed-Meta has better generalization, which is suitable for new unseen tasks. And the target dimension of the Centralized method increases sharply, which may classify similar pictures written in different languages into one category. The experiment result can prove that meta-learning is necessary to deal with this setting.

In Fed-meta, a central server is responsible for aggregating the local model parameter vectors sent by all training nodes without verifying and filtering them. The accuracies of our framework DFMLF-Random and DFMLF-Efficient are both higher than that of Fed-meta, especially on the 10-way 2-shot task based on the MLP model, the accuracies are about 4% higher than Fed-meta. This is because the cross-validation mechanism of our framework can improve the performance of model training.

Compared with DFMLF-Optimal, DFMLF-Random and DFMLF-Efficient are slightly better based on the CNN model, which are about 2% higher based on the MLP model. This is because DFMLF-Optimal leads the system towards the initial committee nodes, resulting in poor generalization performance.

Comparing DFMLF-Random and DFMLF-Efficient, we find that DFMLF-Efficient can achieve similar performance as DFMLF-Random, and on the tasks with only one sample, DFMLF-Efficient is slightly better. This shows that although

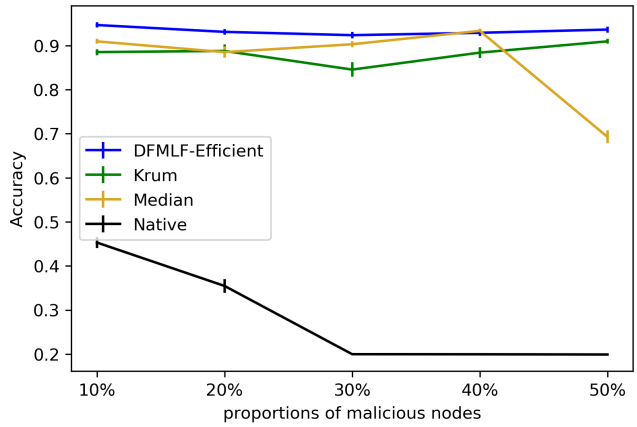

FIGURE 9 The accuracy of the comparison methods under different proportions of malicious nodes

DFMLF-Efficient loses more information than DFMLF-Random, the loss is so low that it can be ignored.

In summary, DFMLF-Efficient can effectively deal with few-shot multi-task problems, and the cross-validation mechanism of our framework can improve the performance of model training. In addition, the random-based committee election strategy has better generalization than the optimal-based committee election strategy, and the loss of the communication-efficient method can be ignored.

4.3.2 | Robustness

In this experiment, we consider the setting of malicious attacks. Due to intentional destruction or transmission error, some training nodes may submit malicious model parameter vectors. If these malicious model parameter vectors are aggregated into the global model, the performance of the global model will be affected. We set the proportion of malicious nodes to 10%, 20%, 30%, 40%, 50%, and ensure that the malicious nodes participating in each round of training are consistent with the overall proportion. We examine a widely-used attack, Random updates [52], in our threat model. When malicious nodes submit model parameter vectors, Gaussian noise (loc=0, scale=0.5) will be added to the vectors. The noise distribution is close to the parameters of the normal vectors, which is difficult to distinguish and has certain concealment. The training setting is the task 5-shot 1-way based on the CNN model. Fig. 9 shows the accuracy of the comparison methods under different proportions of malicious nodes. The short vertical line at each turning point represents the standard deviation.

In Fig. 9, the Naive method scores worst. When the proportion of malicious nodes is 10%, the model effect is severely affected; and when the proportion of malicious nodes exceeds 30%, the effect of the model is the same as that of the random method. This is because the Naive method directly averages all vectors to generate the global model parameter vector without identifying malicious updates. This shows that the malicious nodes have a disastrous effect on the global model, and averaging cannot tolerate a single Byzantine worker.

As for the Median method, when the proportion of malicious nodes is 40% or less, it is robust to malicious attacks; while when the proportion of malicious nodes is up to 50%, the performance of the global model decreases greatly, indicating that the global model aggregates some malicious vectors. This is because when the number of malicious nodes increases, they may collude with each other to make the median deviate from the correct position, and help one

malicious node to be selected.

Krum selects the aggregated model parameter vectors based on the distance of the vectors, while DFMLF-Efficient identifies the honest nodes according to the effect of the model parameter vectors. DFMLF-Efficient and Krum can both achieve stable model performance under different proportions of malicious nodes, which proves the robustness of the two methods. However, our method DFMLF-Efficient is more interpretable, because there is no direct correlation between the distance of the model parameter vectors and the model performance. In addition, the performance of DFMLF-Efficient is better than Krum, this is because the framework based on the committee mechanism is more suitable for meta training, and the cross-validation mechanism can improve the performance of model training. Moreover, in section 3.51, we analyze the computation complexity and point out that DFMLF-Efficient is more efficient than Krum.

4.3.3 | Efficiency

In this experiment, we measure the actual convergence on the task 10-way 2-shot based on the MLP model. As the same as section 4.3.1, we set the number of training nodes participating in each round to 10, and the number of committee nodes is 4. For the setup, we simulate 14 nodes, and each node has the same computing power and transmission capacity. In order to simulate different transmission speeds in different network environments, we set several different transmission speeds: 1Mps, 2Mps, and 5Mps. Under each transmission speed, we evaluate the methods using wall-clock time, which includes computation time and communication time. Fig. 10 shows the test accuracy over wall-clock time for different transmission speeds.

From Fig. 10, we can see that although our framework DFMLF and DFMLF-Efficient product additional computation and communication overhead compared with MLP, their computation overhead is lower than other secure aggregation method Fed-Krum, and their communication overhead is lower than other decentralized method BrainTorrent. The time required for our framework is not much different from that of Fed-Meta. Moreover, with the increase of transmission speed, the advantage of our method DFMLF-Efficient becomes greater. Because DFMLF-Efficient can synchronize training and aggregation, and reduce the waiting time of computation.

Through experimental results and analysis, it can be proved that DFMLF-Efficient can effectively deal with few-shot multi-task problems, and improve the performance of model training. In addition, it can effectively ensure the robustness of model training in the setting of malicious attacks. Moreover, it is more efficient than other secure aggregation methods and other decentralized methods. Compared with the centralized training, DFMLF-Efficient not only considers the attack of malicious nodes or server, but also considers the privacy protection of original data while ensuring the training effect, thus it behaves more applicable in real settings.

4.4 | Hyperparameter Analysis

There are two main hyperparameters that affect the effect of DFMLF: the number of nodes participating in training in each round N_{train}, and the number of committee nodes N_{comm}. By setting a different number of committee nodes and training nodes, we explore the performance of DFMLF. For simplicity, we assume that the number of selected aggregated nodes N is the same as the number of the committee nodes N_{comm}.

The number of committee nodes N_{comm} should be less than the number of training nodes N_{train}. If $N_{train} <= N_{comm}$, all the committee nodes score each training node, resulting in large score differences. It is difficult to select the appropriate training node for aggregation. When $N_{train} > N_{comm}$, the training nodes with better performance will be selected for aggregation, this can not only improve the convergence speed of federated learning, but also improve

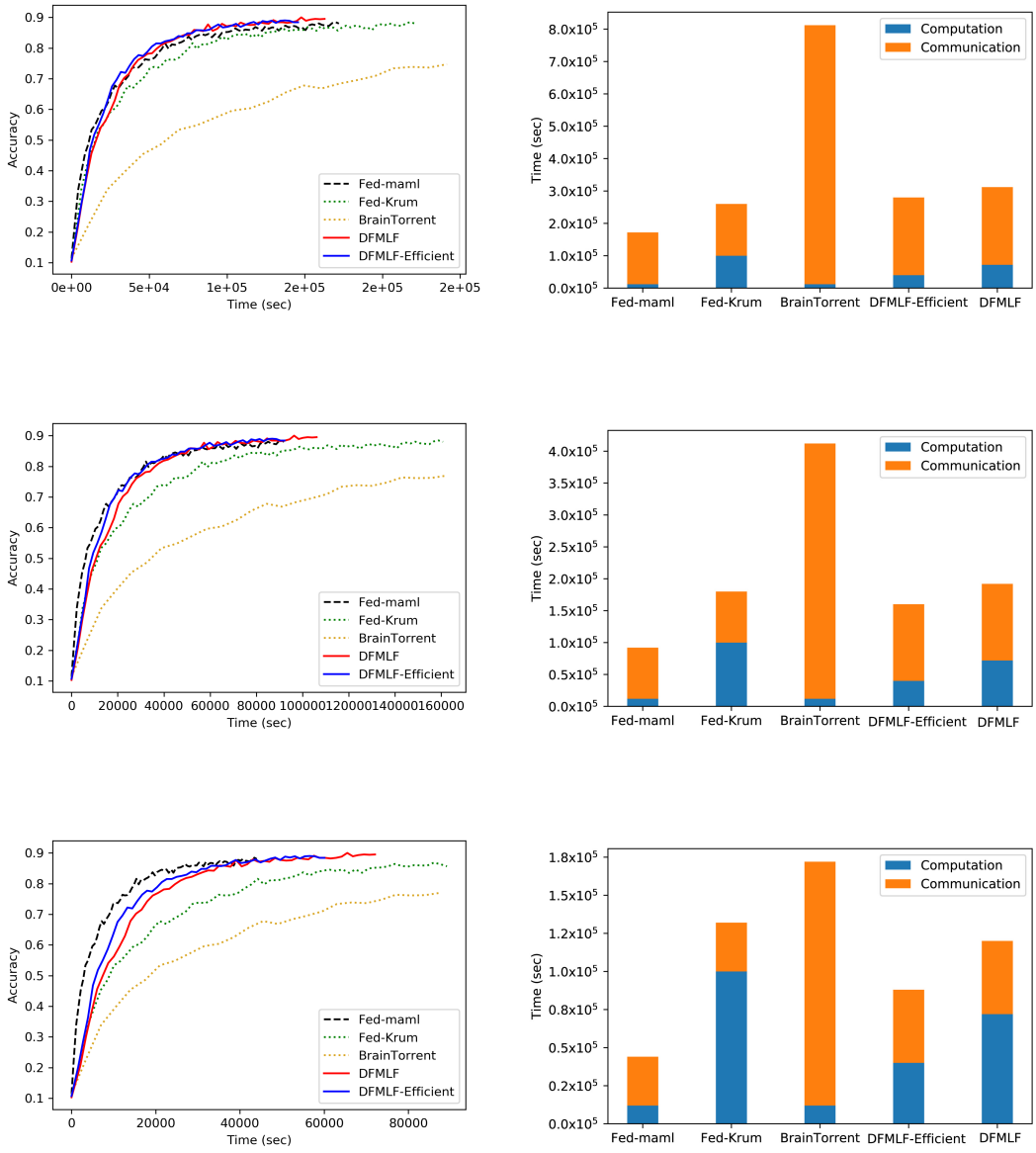

FIGURE 10 The performance over the wall-clock time. Each row shows a different transmission speed. The left column shows convergence via test accuracy vs. wallclock training time; The right column shows the proportion of computation time and communication time.

the performance of the model. However, the number of N_{comm} should be carefully weighed. On the one hand, fewer committee nodes can reduce the workload in the verification phase, as the model parameter vectors of all training

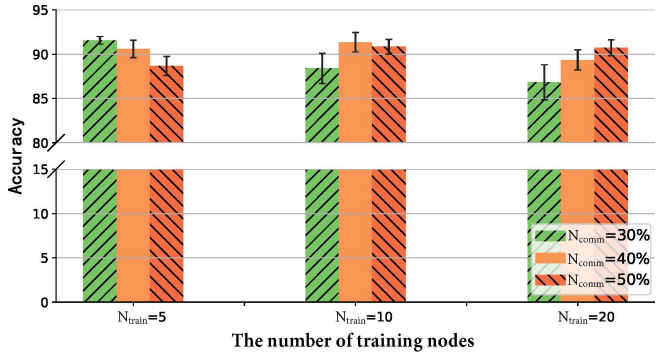

FIGURE 11 The impact of the number of training nodes and committee nodes.

nodes will be submitted to all the committee nodes for verification. On the other hand, an excessively small N_{comm} not only increases the risk of being attacked by malicious nodes, but also leads to too little data in the verification set and insufficient representation of the committee nodes.

In this experiment, the training setting is set to 5-shot, 1-way and the basic model uses the CNN model as Section 4.3.2. In Fig. 11, we show the average accuracy and mark the standard deviation of each result.

We see that when the number of training nodes N_{train} is 5, the best accuracy is achieved when the proportion of committee nodes N_{comm} is 30%; when $N_{train} = 10$, the best N_{comm} is 40%; and when $N_{train} = 20$, the best N_{comm} is 50%. This shows that when N_{train} is relatively small, N_{comm} should also be set smaller, as a larger number of N_{comm} means more scores of the training nodes, and it is difficult to select appropriate training nodes to aggregate due to the greate difference in scores; and when N_{train} is relatively large, N_{comm} also be set larger, as the cross-validation of the committee nodes can improve the performance of model training.

5 | CONCLUSION

Federated learning is increasingly attractive as it provides globally model training while ensuring privacy. To solve the few-shot multi-task problem in a distributed setting, we propose that the devices take the rapid adaptation as the objective and train an excellent initial value of model parameters to deal with different tasks. Considering the malicious attacks of the training nodes and the central server, this paper proposes a server-less training framework. The server-less design can avoid the privacy leakage risk brought by the central server, while the cross-validation and secure aggregation mechanism reduce the impact of malicious devices and improve security. In the experiment, we construct several different task settings using a real-world dataset, and demonstrate the effectiveness, robustness, and efficiency of our framework by comparing it with other frameworks and secure aggregation methods.

However, there are still some shortcomings in this paper. For example, when electing committee nodes, the hardware conditions of the nodes, such as computing power and network bandwidth, are not considered. In the future, we are going to further this study in two aspects. Firstly, we will consider more influencing factors and study election strategies in different scenarios. Secondly, we will pay attention to the improvement of transmission efficiency, the optimization of storage efficiency.

CONFLICT OF INTERESTS

The authors declare that there are no conflict of interests.

ACKNOWLEDGEMENTS

The research is supported by the National Key R&D Program of China (2020YFB1006001), the National Natural Science Foundation of China (62176269, 62076179, 61732011), the Guangdong Basic and Applied Basic Research Foundation (2019A1515011043), the Beijing Natural Science Foundation (Z180006) and the Innovative Research Foundation of Ship General Performance (25622112).

REFERENCES

[1] Yang Q, Liu Y, Chen T, Tong Y. Federated Machine Learning: Concept and Applications. ACM Trans Intell Syst Technol 2019;10(2):12:1–12:19.

[2] Konecný J, McMahan HB, Ramage D, Richtárik P. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. CoRR 2016;abs/1610.02527.

[3] Konecný J. Stochastic, Distributed and Federated Optimization for Machine Learning. CoRR 2017;abs/1707.01155.

[4] McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA. Communication-Efficient Learning of Deep Networks from Decentralized Data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, Fort Lauderdale, USA, vol. 54 PMLR; 2017. p. 1273–1282.

[5] Lyu L, Yu H, Zhao J, Yang Q. Threats to Federated Learning. In: Federated Learning - Privacy and Incentive, vol. 12500 of Lecture Notes in Computer Science Springer; 2020.p. 3–16.

[6] Kang J, Xiong Z, Li X, Zhang Y, Niyato D, Leung C, et al. Optimizing Task Assignment for Reliable Blockchain-Empowered Federated Edge Learning. IEEE Trans Veh Technol 2021;70(2):1910–1923.

[7] Finn C, Abbeel P, Levine S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In: Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, vol. 70; 2017. p. 1126–1135.

[8] Schmidhuber J. Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook. PhD thesis, Technische Universität München; 1987.

[9] Finn C, Xu K, Levine S. Probabilistic Model-Agnostic Meta-Learning. In: Annual Conference on Neural Information Processing Systems, Montréal, Canada; 2018. p. 9537–9548.

[10] Chen F, Luo M, Dong Z, Li Z, He X. Federated meta-learning with fast convergence and efficient communication. arXiv preprint arXiv:180207876 2018;.

[11] Hospedales TM, Antoniou A, Micaelli P, Storkey AJ. Meta-Learning in Neural Networks: A Survey. CoRR 2020;abs/2004.05439.

[12] Finn C, Rajeswaran A, Kakade SM, Levine S. Online Meta-Learning. In: Chaudhuri K, Salakhutdinov R, editors. Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, USA, vol. 97 of Proceedings of Machine Learning Research PMLR; 2019. p. 1920–1930.

[13] Fung C, Yoon CJM, Beschastnikh I. Mitigating Sybils in Federated Learning Poisoning. CoRR 2018;abs/1808.04866.

[14] Gu T, Liu K, Dolan-Gavitt B, Garg S. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. IEEE Access 2019;7:47230–47244.

[15] Bhagoji AN, Chakraborty S, Mittal P, Calo SB. Analyzing Federated Learning through an Adversarial Lens. In: Chaudhuri K, Salakhutdinov R, editors. Proceedings of the 36th International Conference on Machine Learning, ICML, Long Beach, California, USA, vol. 97 of Proceedings of Machine Learning Research PMLR; 2019. p. 634–643.

[16] Bagdasaryan E, Veit A, Hua Y, Estrin D, Shmatikov V. How To Backdoor Federated Learning. In: The 23rd International Conference on Artificial Intelligence and Statistics, Online [Palermo, Sicily, Italy], vol. 108 PMLR; 2020. p. 2938–2948.

[17] Blanchard P, Mhamdi EME, Guerraoui R, Stainer J. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA; 2017. p. 119–129.

[18] Kang J, Xiong Z, Niyato D, Xie S, Zhang J. Incentive Mechanism for Reliable Federated Learning: A Joint Optimization Approach to Combining Reputation and Contract Theory. IEEE Internet Things J 2019;6(6):10700–10714.

[19] Pappas C, Chatzopoulos D, Lalis S, Vavalis M. IPLS: A Framework for Decentralized Federated Learning. In: IFIP Networking Conference, IFIP Networking 2021, Espoo and Helsinki, Finland, June 21-24, 2021 IEEE; 2021. p. 1–6.

[20] Hu C, Jiang J, Wang Z. Decentralized Federated Learning: A Segmented Gossip Approach. CoRR 2019;abs/1908.07782.

[21] Roy AG, Siddiqui S, Pölsterl S, Navab N, Wachinger C. BrainTorrent: A Peer-to-Peer Environment for Decentralized Federated Learning. CoRR 2019;abs/1905.06731.

[22] Li Y, Chen C, Liu N, Huang H, Zheng Z, Yan Q. A Blockchain-Based Decentralized Federated Learning Framework with Committee Consensus. IEEE Netw 2021;35(1):234–241.

[23] Ng JS, Lim WYB, Xiong Z, Cao X, Jin J, Niyato D, et al. Reputation-aware Hedonic Coalition Formation for Efficient Serverless Hierarchical Federated Learning. IEEE Trans Parallel Distributed Syst 2021;p. 1–1.

[24] Lim WYB, Ng JS, Xiong Z, Jin J, Zhang Y, Niyato D, et al. Decentralized Edge Intelligence: A Dynamic Resource Allocation Framework for Hierarchical Federated Learning. IEEE Trans Parallel Distributed Syst 2022;33(3):536–550.

[25] Yan H, Hu L, Xiang X, Liu Z, Yuan X. PPCL: Privacy-preserving collaborative learning for mitigating indirect information leakage. Inf Sci 2021;548:423–437.

[26] Hu L, Yan H, Li L, Pan Z, Liu X, Zhang Z. MHAT: An efficient model-heterogenous aggregation training scheme for federated learning. Inf Sci 2021;560:493–503.

[27] Wang X, Liang Z, Koe ASV, Wu Q, Zhang X, Li H, et al. Secure and efficient parameters aggregation protocol for federated incremental learning and its applications. International Journal of Intelligent Systems 2021;p. 1–17.

[28] Humbeck L, Morawietz T, Sturm N, Zalewski A, Harnqvist S, Heyndrickx W, et al. Don't Overweight Weights: Evaluation of Weighting Strategies for Multi-Task Bioactivity Classification Models. Molecules 2021;26(22).

[29] Smith V, Chiang C, Sanjabi M, Talwalkar AS. Federated Multi-Task Learning. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA; 2017. p. 4424–4434.

[30] Mills J, Hu J, Min G. Multi-Task Federated Learning for Personalised Deep Neural Networks in Edge Computing. IEEE Trans Parallel Distributed Syst 2022;33(3):630–641.

[31] Li R, Ma F, Jiang W, Gao J. Online Federated Multitask Learning. In: 2019 IEEE International Conference on Big Data (IEEE BigData), Los Angeles, CA, USA, December 9-12, 2019 IEEE; 2019. p. 215–220.

[32] Corinzia L, Buhmann JM. Variational Federated Multi-Task Learning. CoRR 2019;abs/1906.06268.

[33] Dinh CT, Vu TT, Tran NH, Dao MN, Zhang H. FedU: A Unified Framework for Federated Multi-Task Learning with Laplacian Regularization. CoRR 2021;abs/2102.07148.

[34] He C, Ceyani E, Balasubramanian K, Annavaram M, Avestimehr S. SpreadGNN: Serverless Multi-task Federated Learning for Graph Neural Networks. CoRR 2021;abs/2106.02743.

[35] Jiang Y, Konecný J, Rush K, Kannan S. Improving Federated Learning Personalization via Model Agnostic Meta Learning. In: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada; 2019. p. 1–12.

[36] Lin S, Yang L, He Z, Fan D, Zhang J. MetaGater: Fast Learning of Conditional Channel Gated Networks via Federated Meta-Learning 2021;p. 164–172.

[37] Fallah A, Mokhtari A, Ozdaglar AE. Personalized Federated Learning: A Meta-Learning Approach. CoRR 2020;abs/2002.07948.

[38] Yue S, Ren J, Xin J, Zhang D, Zhang Y, Zhuang W. Efficient Federated Meta-Learning over Multi-Access Wireless Networks. CoRR 2021;abs/2108.06453.

[39] Aramoon O, Chen P, Qu G, Tian Y. Meta Federated Learning. CoRR 2021;abs/2102.05561.

[40] Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, et al. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA ACM; 2017. p. 1175–1191.

[41] Castro M, Liskov B. Practical Byzantine Fault Tolerance and Proactive Recovery. ACM Trans Comput Syst 2002 11;20:398–461.

[42] Blanchard P, Mhamdi EME, Guerraoui R, Stainer J. Byzantine-Tolerant Machine Learning. CoRR 2017;abs/1703.02757.

[43] Chen Y, Su L, Xu J. Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent. In: Psounis K, Akella A, Wierman A, editors. Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS 2018, Irvine, CA, USA, June 18-22, 2018 ACM; 2018. p. 96.

[44] McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA. Communication-Efficient Learning of Deep Networks from Decentralized Data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, vol. 54 PMLR; 2017. p. 1273–1282.

[45] Li Y, Yu M, Li S, Avestimehr S, Kim NS, Schwing AG. Pipe-SGD: A Decentralized Pipelined SGD Framework for Distributed Deep Net Training. In: Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada; 2018. p. 8056–8067.

[46] Yang C, Wang Q, Xu M, Chen Z, Bian K, Liu Y, et al. Characterizing Impacts of Heterogeneity in Federated Learning upon Large-Scale Smartphone Data. In: WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021 ACM / IW3C2; 2021. p. 935–946.

[47] Vinyals O, Blundell C, Lillicrap T, Kavukcuoglu K, Wierstra D. Matching Networks for One Shot Learning. In: Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain; 2016. p. 3630–3638.

[48] Chen C, Golubchik L, Paolieri M. Backdoor Attacks on Federated Meta-Learning. CoRR 2020;abs/2006.07026.

[49] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016 IEEE Computer Society; 2016. p. 770–778.

[50] Yagli S, Dytso A, Poor HV. Information-Theoretic Bounds on the Generalization Error and Privacy Leakage in Federated Learning. In: 21st IEEE International Workshop on Signal Processing Advances in Wireless Communications IEEE; 2020. p. 1–5.

[51] Cao T, Truong-Huu T, Tran HD, Tran K. A Federated Learning Framework for Privacy-preserving and Parallel Training. CoRR 2020;abs/2001.09782.

[52] Xu X, Lyu L. Towards Building a Robust and Fair Federated Learning System. CoRR 2020;abs/2011.10464.

XIAOLI LI is currently working toward the PhD degree in the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. She received the Master's degree in computer architecture from University of Electronic Science and Technology of China, Chengdou, China, in 2011. Her research interests include services computing, software engineering, cloud computing, machine learning and federated learning.

YUZHENG LI received the B.E. degrees from the Sun Yat-sen University, Guangdong, China, in 2018. He is currently a M.S. student of Sun Yat-sen University. His current research interests include federated learning, blockchain, statistical machine learning, multi-view learning and optimization.

JINING WANG received the B.E. degrees from the Sun Yat-sen University, Guangdong, China, in 2020. He is currently a M.S. student of Sun Yat-sen University. His current research interests include federated learning, knowledge graph, graph neural network, and causal inference.

CHUAN CHEN received the B.S. degree from Sun Yat-sen University, Guangzhou, China, in 2012, and the Ph.D. degree from Hong Kong Baptist University, Hong Kong, in 2016. He is currently an Associate Professor with the School of Computer Science and Engineering, Sun Yat-Sen University. He published over 50 international journal and conference papers. His current research interests include machine learning, numerical linear algebra, and numerical optimization.

LIU YANG received the Ph.D. degree in computer science from the School of Computer and Information Technology, Beijing Jiaotong University, China, in 2016. She is currently an Associate Professor with the College of Intelligence and Computing, Tianjin University. Her research interests include data mining and machine learning.

ZIBIN ZHENG received his PhD degree from the Chinese University of Hong Kong in 2011. He is currently a professor in the School of Computer Science and Engineering at Sun Yat-sen University, China. He has published over 150 international journal and conference papers, including three ESI highly cited papers. According to Google Scholar, his papers have more than 13,590 citations, with an H-index of 54. His research interests include blockchain, smart contract, services computing, software reliability.