A Decentralized Federated Learning Framework via Committee Mechanism with Convergence Guarantee

Chunjiang Che, Xiaoli Li, *Student Member, IEEE,*
Chuan Chen, *Member, IEEE,* Xiaoyu He, and Zibin Zheng, *Senior Member, IEEE*

Abstract—Federated learning allows multiple participants to collaboratively train an efficient model without exposing data privacy. However, this distributed machine learning training method is prone to attacks from Byzantine clients, which interfere with the training of the global model by modifying the model or uploading the false gradient. In this paper, we propose a novel serverless federated learning framework *Committee Mechanism based Federated Learning* (CMFL), which can ensure the robustness of the algorithm with convergence guarantee. In CMFL, a committee system is set up to screen the uploaded local gradients. The committee system selects the local gradients rated by the elected members for the aggregation procedure through the selection strategy, and replaces the committee member through the election strategy. Based on the different considerations of model performance and defense, two opposite selection strategies are designed for the sake of both accuracy and robustness. Extensive experiments illustrate that CMFL achieves faster convergence and better accuracy than the typical Federated Learning, in the meanwhile obtaining better robustness than the traditional Byzantine-tolerant algorithms, in the manner of a decentralized approach. In addition, we theoretically analyze and prove the convergence of CMFL under different election and selection strategies, which coincides with the experimental results.

Index Terms—Decentralized Federated Learning, Committee Mechanism, Byzantine Robustness, Theoretical Convergence Analysis.

1 Introduction

Nowadays, data arise from a wide range of sources, such as mobile devices, commercial, industrial and medical activities. These data are further used for training the Artificial Intelligence (AI) models applied in a variety of fields. The conventional AI methods always require uploading the source data to a central server. However, this is usually impractical due to data privacy or commercial competition. Federated Learning (FL) [1], which allows multiple devices to train a shared global model without uploading the local source data to a central server, is an effective way to solve the aforementioned problem. In FL settings, multiple clients (also known as participants) are responsible for model training and uploading the local gradients, while the central server is responsible for the model aggregation. A single round of FL mainly follows the following four steps: (1) the multiple clients download a global model from the server, and train their local models on their local datasets; (2) the clients upload the local gradients to the server, and the server aggregates the received multiple local gradients to construct the global gradient; (3) the server uses the global gradient to update the global model; (4) the clients download the global model to the local to continue the next training round. The above operations will be repeated until the algorithm converges.

It is fascinating that FL can perform model training without uploading source data, McMahan *et al.* [2] proposed that FL can achieve a similar test accuracy as the centralized method based on the full training dataset while providing stronger privacy guarantees. However, Lyu *et al.* [3] proposed that the conventional FL is vulnerable to malicious attacks from the Byzantine clients and the central server. For example, the Byzantine clients upload false gradients to affect the performance of the global model, which may lead to training failure [4] [5]. Besides, the presence of malicious server is widely considered in FL [6] [7] [8] [9], and Hu *et al.* [10] proposed that external attacks on the central server will cause the entire learning process to terminate. In recent years, there have been a lot of works to solve the security problem of FL. Some works aim to design a robust aggregation rule to reduce the negative impact of malicious gradients [11]. For example, Blanchard *et al.* [4] proposed a Byzantine-tolerant algorithm Krum, which can tolerate Byzantine workers via aggregation rule with resilience property. Similarly, Yin *et al.* [12] proposed two robust distributed gradient descent algorithms based on coordinate-wise median and trimmed mean operations for Byzantine-robust distributed learning. Chen *et al.* [13] proposed a simple variant of the classical gradient descent method based on the geometric median of means of the gradients, which can tolerate the Byzantine failures. Some other works detect Byzantine attackers and remove the malicious local gradients from aggregation by clustering [14] [15]. Different from the aforementioned works, Li *et al.* [16] propose a framework that trains a detection model on the server to detect and remove the malicious local gradients. Although they can guarantee the convergence and Byzantine-tolerant, they cannot provide effective defense in the presence of malicious servers [17].

Chunjiang Che, Xiaoli Li, Chuan Chen, and Xiaoyu He are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. Zibin Zheng is the School of Software Engineering, Sun Yat-sen University, Zhuhai, China. e-mail: {chechj, lixli27}@mail2.sysu.edu.cn, {chenchuan, hexy73, zhzibin}@mail.sysu.edu.cn. Corresponding author: Chuan Chen.

As for the further comment, the typical FL requires a central server to complete the gradient aggregation procedure, thus it is difficult to find a fully trusted server in actual scenarios. Beyond that, the entire FL system will be paralyzed if the server suffers a malicious attack. Therefore, a lot of works are devoted to designing a serverless FL framework to reduce the risk of single point failure [18]. Some existing works design serverless FL frameworks by learning from network protocols such as P2P [19] [20] and Gossip [21] [22]. These approaches treat clients as network nodes, which communicate with each other according to the improved network protocol and complete the local training and the aggregation of the global model. Besides, other approaches employ blockchain technology to complete the work of the server to achieve a serverless FL framework [23] [24] [25] [26]. They treat clients as blockchain nodes, record the local gradients uploaded on the block, and then make the leading who completes the PoW (Proof of Work) to ensure the aggregation procedure. However, few works present the theoretical convergence analysis of the serverless FL, leading to the lack of performance guarantee.

In this paper, we comprehensively consider Byzantine attacks of both clients and the central server, and design a serverless Federated Learning framework under Committee Mechanism (CMFL), in which some participating clients are appointed as committee members, and the committee members take the responsibility for monitoring the entire FL training process and ensuring the dependable aggregation of the local gradients. The CMFL consists of three components: the scoring system, the election strategy and the selection strategy. The scoring system plays a role in distinguishing different clients. The election strategy is responsible for selecting some clients who can represent the majority of clients to become members of the new committee. Based on different considerations, we propose two opposite selection strategies to make the committee members verify local gradients uploaded by clients. A selection strategy is designed to ensure the robustness of the training process, where the committee members accept those local gradients who are similar to their own local gradients but reject those local gradients who are significantly different from their own ones. The other selection strategy is designed to accelerate the convergence of the global model in a non-attack scenario, where the committee members accept those local gradients who are different from their own local gradients but reject those gradients who are similar to their own ones. Compared to some existing Byzantine-tolerant algorithms based on local gradient validation, such as Median [12], Trimmed Mean [12] and Krum [4], CMFL achieves better performance and higher robustness over various datasets, illustrated by the experimental results. Besides, CMFL can achieve a higher security level due to its decentralized setting. Theoretical analysis on the convergence of the model is further presented for the performance guarantee, which illustrates the impact of the proposed election and selection strategies. Extensive experiments further demonstrate the outperformance of CMFL compared with both the typical and Byzantine-tolerant FL models, coinciding with the theoretical analysis on the efficiency of the proposed election and selection strategies. In summary, we highlight the contributions as follow:

- We propose a serverless FL framework: CMFL, with the ability to monitor the gradient aggregation procedure and prevent both malicious clients and the server from hindering the training of the global model.
- We propose an election strategy and two selection strategies suitable for different scenarios, which can ensure the robustness of the algorithm or accelerate the training process.
- We give the proof and analysis of the convergence of the proposed serverless FL framework for the theoretical guarantee, which considers the influence of election and selection strategies on the performance of the global model.
- We conduct extensive experimental results to show that CMFL has a faster model convergence rate and better model performance than the typical and Byzantine-tolerant FL models.

The remainder of this paper is organized as follows. Section 2 surveys related work. Section 3 briefly outlines the background and problem formulation of FL. Section 4 introduces our proposed framework. We show the convergence analysis in Section 5.1 and complexity analysis in Section 5.2. Experimental results and analysis are summarized in Section 6. We conclude the paper in Section 7. Finally, Supplement gives the convergence proof.

2 Related Work

Konečný *et al.* [27] introduce a new distributed optimization setting in machine learning in 2015. Based on this setting the concept of FL is proposed [1], which aims to train an efficient centralized model in a scenario where the training data is distributed across a large number of clients. However, these frameworks suffer from heterogeneous clients, failing to achieve satisfactory performance on the Non-IID dataset. To further handle such issues, several works extend FL to Non-IID dataset. Li *et al.* [28] presents the theoretical guarantees under Non-IID settings and analyzes the convergence of FedAvg. Li *et al.* [29] propose a framework *FedProx* with convergence guarantees to tackle heterogeneity on the Non-IID dataset. Yue *et al.* [30] proposed a strategy to mitigate the negative impact of Non-IID data by share a small subset of data between all the clients. Briggs *et al.* [31] improve the performance of FL on the Non-IID dataset by introducing a hierarchical clustering step to separate clusters of clients. For the study on data heterogeneity, Haddadpour *et al.* [32] generalize the local stochastic and full gradient descent with a new scheme, periodic averaging, to solve nonconvex optimization problems in FL. Dinh *et al.* [33] proposed FEDL, a FL algorithm which can handle heterogeneous user data without any assumptions except strongly convex and smooth loss functions. Liu *et al.* [34] proposed momentum federated learning (MFL) to accelerate the convergence, and they establish global convergence properties of MFL and originate an upper bound on the convergence rate. Different from the aforementioned works, the proposed framework provides a new perspective to enhance the performance on the Non-IID setting.

The attractiveness of Federated learning relies on the trainable centralized model on user equipment without uploading user data. However, such a framework is vulnerable to Byzantine attackers due to the lack of identity authentication for local gradients. Lots of works have designed a series of Byzantine-tolerant algorithms to further ensure the robustness of the training process. For example, Blanchard *et al.* [4] proposed *Krum*, which aims to select the global model update based on the Euclidean distance between the local models. Yin *et al.* [12] proposed *Median* and *Trimmed Mean*, which are designed to remove extreme local gradients to ensure the robustness of the algorithm. The Median method constructs a global gradient, where each element is the median of the elements in the local gradients with the same coordinate, while the Trimmed Mean method removes the maximum and minimum fraction of elements in the local gradients, and then performs a weighted average on the remaining local gradients to construct the global gradient. Muñoz-González *et al.* [11] propose a Hidden Markov Model to learn the quality of local gradients and design a robust aggregation rule, which discards the bad local gradients in the aggregation procedure. The aforementioned algorithms are all trying to ensure the robustness of FL by designing a more appropriate aggregation mechanism. However, these works can not provide effective defense in the presence of malicious servers.

In real applications, commercial competition makes it difficult to find a fully trusted central server among the participants. In addition, server error or malicious server will also cause irreparable damage to the FL system. In this way, many serverless FL frameworks are proposed to solve these problems. Among these approaches, some of them achieve a serverless FL framework by imitating existing network protocols. For example, Abhijit *et al.* [19] proposed a P2P serverless FL framework, where any two clients exchange information end-to-end and update their local models at each epoch. Hu *et al.* [21] used the Gossip protocol to complete the model aggregation process, which takes on the role of the central server. Besides, other works build a serverless FL framework based on the blockchain system. For example, Kim *et al.* [23] proposed a blockchain FL architecture, in which they divide the blockchain nodes into devices and miners. The device nodes provide data, train the model locally, and upload the local gradients to their associated miner in the blockchain network. Miner nodes exchange and verify all the local gradients. Although these works have obtained the corresponding performance to some degree, they lack the theoretical analysis of the model convergence under serverless FL framework and consideration of Byzantine attacks of clients.

To deal with the limitations of the existing works, we proposed a serverless FL framework under committee mechanism. In scenarios that consider the Byzantine attacks, the framework can be efficient Byzantine-tolerant of malicious clients and servers. In scenarios without considering the Byzantine attacks, the framework can achieve better performance of the global model on the Non-IID dataset. Beyond that, we present the theoretical analysis of the convergence under the framework.

3 Background

3.1 Federated Learning

A typical FL framework consists of a central server and multiple clients. The server maintains a global model, and each client maintains a local model. At the beginning of training, the global model and all local models will be initialized randomly. And then the following steps will be performed at each communication round to continue the training process [1]:

1. The server randomly selects a subset of clients, which then download the global model to the local.
2. Each client in the subset performs a certain number of Stochastic Gradient Descent (SGD) [35] [36] and computes the local gradient.
3. The clients in the subset send their local gradients to the server.
4. The server receives the local gradients and performs the Federated Averaging (FedAvg) algorithm [37] to construct a global gradient, which is used to update the global model.

The above steps will be iterated until the algorithm converges or the model accuracy meets the requirements.

3.2 Problem Formulation

We consider the typical FL setup with total K clients. The k-th client for $k = 1, ..., K$ owns a local dataset $\mathcal{D}_k$, which contains $n_k = |\mathcal{D}_k|$ data samples. We denote the user-defined loss function for sample ξ and model parameter vector $\mathbf{w}$ as $f(\mathbf{w}, \xi)$, the local objective function $F_k(\mathbf{w})$ on the k-th client can be written as follows:

$$F_k(\mathbf{w}) = \frac{1}{n_k} \sum_{\xi \in \mathcal{D}_k} f(\mathbf{w}, \xi). \tag{1}$$

We consider the following global objective function:

$$F(\mathbf{w}) = \sum_{k=1}^{K} p_k F_k(\mathbf{w}), \tag{2}$$

where $p_k = n_k / \sum_{k=1}^{K} n_k$ denotes the weight of the dataset on the k-th client. Formally, $\nabla F_k(\mathbf{w}_{k,i}^t)$ denotes the local gradient over dataset $\mathcal{D}_k$. Assume that at round t the k-th client trains its local model $\mathbf{w}_{k,i}^t$ over mini-batch $\mathcal{B}_{k,i}^t$ for i iterations of SGD, where $\mathcal{B}_{k,i}^t$ is randomly sampled from $\mathcal{D}_k$. Then the k-th client computes the local gradient $g_k(\mathbf{w}_{k,i}^t, \mathcal{B}_{k,i}^t)$ by the following formula:

$$g_k(\mathbf{w}_{k,i}^t, \mathcal{B}_{k,i}^t) = \frac{1}{|\mathcal{B}_{k,i}^t|} \sum_{\xi \in \mathcal{B}_{k,i}^t} \nabla f(\mathbf{w}_{k,i}^t, \xi). \tag{3}$$

The $g_k(\mathbf{w}_{k,i}^t, \mathcal{B}_{k,i}^t)$ is used to update the local model $\mathbf{w}_{k,i}^t$ as follows:

$$\mathbf{w}_{k,i+1}^t = \mathbf{w}_{k,i}^t - \eta_i^t g_k(\mathbf{w}_{k,i}^t, \mathcal{B}_{k,i}^t), \tag{4}$$

where η_i^t represents the local learning rate at iteration i of round t and τ represents the maximal local SGD iterations. And after τ iterations of local updating, the local gradient $g_k(\mathbf{w}_{k,\tau}^t, \mathcal{B}_{k,\tau}^t)$ is sent to the server to construct a global gradient as follows:

$$\overline{g}^t = \sum_{k \in S^t} p_{k,S^t} g_k(\mathbf{w}_{k,\tau}^t, \mathcal{B}_{k,\tau}^t), \tag{5}$$

where S^t denotes the subset of clients and $p_{k,S^t} = n_k / \sum_{k \in S^t} n_k$ is the weight of the dataset on the k-th client of S^t. The $\overline{\mathbf{w}}^t$ is updated at each round as follows:

$$\overline{\mathbf{w}}^{t+1} = \overline{\mathbf{w}}^t - \eta_t \overline{g}^t, \tag{6}$$

where η_t represents the global learning rate.

4 Committee Mechanism based Federated Learning

The typical FL system is vulnerable to Byzantine attacks and malicious servers due to its disability to implement a serverless framework and the lack of verification for the uploaded local gradients. The key insight of CMFL is to utilize the committee mechanism to implement a decentralized framework. Under such a decentralized framework, we appoint some training clients as the committee members, which are responsible for filtering the local gradients uploaded by the remaining clients. The committee members must reach a consensus on which the uploaded gradient should be used for the global updating. In this way, the aggregation process is controlled by all the committee members rather than one untrustworthy central server. As long as the number of the honest committee members is greater than that of the malicious members, attacks from malicious clients become meaningless, since the decision of the committee depends on the majority of the committee members. In order to guarantee the honesty of committee members and achieve a secure aggregation, we design a novel committee mechanism, including a scoring system, selection strategy, election strategy, and committee consensus protocol. The detailed introduction of the proposed framework and the training progress is involved in this section and the further theoretical analysis of the framework is illustrated in Section 5.1.

4.1 Framework of CMFL

In the CMFL framework shown in Figure 1, the clients are divided into three categories: *training client*, *committee client*, and *idle client*. At each round, the following steps are performed to complete the training process:

- **Activate.** A part of the idle clients are activated to be the training clients, which participate in the training at this round.
- **Training.** The training clients and the committee clients download the global model to the local for training, while the idle clients stay idle until the next round. The training clients and the committee clients perform SGD over their local dataset and compute the local gradients. The difference is that the local gradients on the training clients are used to update the global model, while the local gradients on the committee clients are used to verify the gradients uploaded by the training clients.
- **Scoring.** The training clients send their local gradients to each committee client and the committee clients assign a score on them according to an established scoring system.
- **Selection.** Only the qualified gradients according to the set selection strategy will be used to construct the global gradient.
- **Aggregation.** The clients who meet the selection strategy are responsible for completing the aggregation procedure, which is called the *aggregation client*.
- **Election.** An election strategy is designed to complete the replacement of committee members. A part of the training clients who meet the election strategy become the new committee clients.
- **Step Down.** The prior committee clients become the idle clients, waiting for the next participate.

In a decentralized framework, we design a committee consensus protocol based on Practical Byzantine Fault Tolerance(pBFT) [38], to complete the Selection, Aggregation, and Election process.

Supposed that C clients will be selected as committee clients, the local model of c-th committee client for $c = 1, ..., C$ at round t is expressed as $\mathbf{w}_{c,\tau}^t$. Assume that we have a total of K clients, and m gradients will be accepted at each round, which are verified by C committee members. Also, we represent the committee client set, the training client set and the aggregation client set at round t as S_c^t, S_b^t and S_a^t respectively. The relevant symbols are shown in Table 1.

TABLE 1: Notation

Notation	Description
K	Number of total clients
T	Number of maximal communication rounds
τ	Number of maximal SGD iterations
$\mathcal{D}_k$	Local dataset
$\mathcal{B}_{k,i}^t$	Mini-batch randomly sampled from $\mathcal{D}_k$
$\mathbf{w}_{k,i}^t$	Local model parameter
$\overline{\mathbf{w}}^t$	Global model parameter
$\mathbf{w}_k^*$	Optimal local model parameter
$\mathbf{w}^*$	Optimal global model parameter
$\mathbf{w}_{c,i}^t$	Local model parameter of committee client
$f(\mathbf{w}, \xi)$	Overall loss function
$F_k(\mathbf{w})$	Local objective function
$g_k(\mathbf{w}_{k,i}^t, \mathcal{B}_{k,i}^t)$	Mini-batch local gradient for i iteration
$\nabla F_k(\mathbf{w}_{k,i}^t)$	Full local gradient for i iteration
$\hat{g}_k^t$	Local gradient for τ iterations
F_k^*	Optimal value of the local objective function
F^*	Optimal value of the global objective function
S_a^t	Aggregation client set
S_b^t	Training client set
S_c^t	Committee client set
m	Number of clients in S_a^t
n	Number of clients in S_b^t
C	Number of clients in S_c^t
$\mathcal{P}_k^c$	Score of the k-th client assigned by the c-th client
$\mathcal{P}_k$	Final score of the k-th client

4.2 Committee Mechanism

A committee system is set up to complete the screening of local gradients. The committee system consists of the scoring system, the election strategy, the selection strategy and the committee consensus protocol.

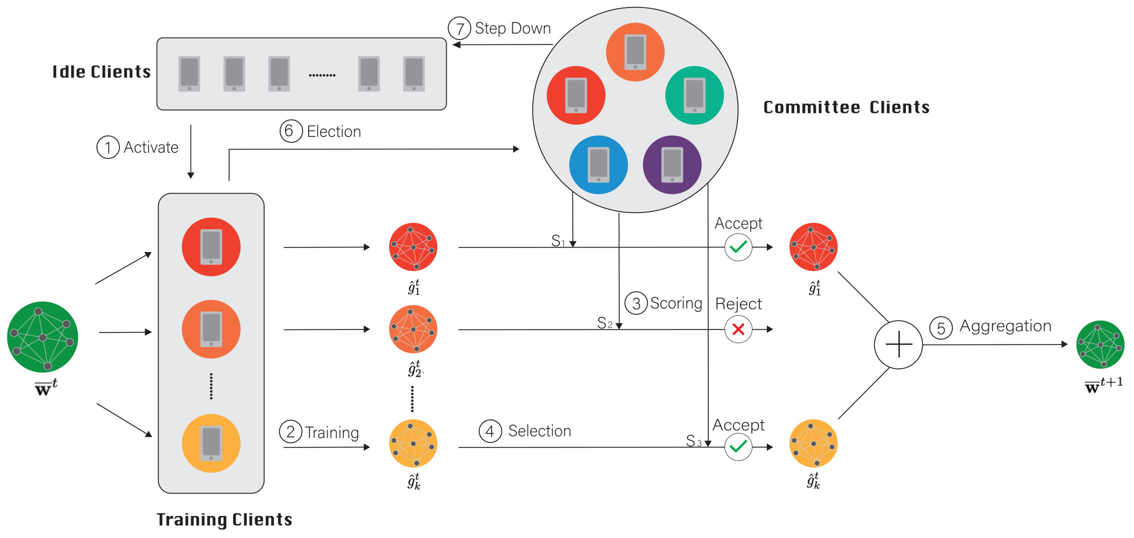


Fig. 1: The training process of CMFL are as follows: (1) Training clients are selected randomly from the idle clients; (2) The training clients and committee clients download the global model $\overline{\mathbf{w}}^t$ and start training, then send their local gradients $\hat{g}_k^t$ to the committee clients; (3) The committee client assigns a score to each client according to the scoring system; (4) The committee selects the aggregation clients according to the selection strategy; (5) The local gradients uploaded by the aggregation clients are used to construct the global gradient $\overline{\mathbf{w}}^{t+1}$; (6) New committee clients are elected from the training clients according to the election strategy; (7) The committee clients of the last round are added to the idle client list.

4.2.1 *Scoring System*

The key insight of the scoring system is to distinguish between the honest and malicious gradients by calculating their Euclidean distance. The Euclidean distance of two honest gradients is lower than that of an honest gradient and a malicious gradient. Based on this insight, the scoring system is designed as comparing the Euclidean distance of two gradients, where the clients who upload the honest gradients can obtain higher scores while the clients who upload the malicious gradients will obtain the lower score. Assume that the local gradient on the k-th training client at round t is denoted as $\hat{g}_k^t = g_k(\mathbf{w}_{k,\tau}^t, \mathcal{B}_{k,\tau}^t)$, and the local gradient on the the c-th committee client at round t is expressed as $\hat{g}_c^t = g_c(\mathbf{w}_{c,\tau}^t, \mathcal{B}_{k,\tau}^t)$. The score $\mathcal{P}_k^c$ of the k-th training client assigned by the c-th committee client is computed as follows:

$$\mathcal{P}_k^c = \frac{1}{||\hat{g}_k^t - \hat{g}_c^t||_2^2}. \tag{7}$$

Since $\hat{g}_k^t$ and $\hat{g}_c^t$ are local gradients generated from different clients, we assume that $\hat{g}_k^t \neq \hat{g}_c^t$ for any $k \in S_b^t, c \in S_c^t$. We define

$$\mathcal{P}_k = \frac{1}{\frac{1}{C}\sum_c^C ||\hat{g}_k^t - \hat{g}_c^t||_2^2} = \frac{C}{\sum_c^C \frac{1}{\mathcal{P}_k^c}} \tag{8}$$

as the final score of the k-th training client. The scoring principle is mainly based on the Euclidean distance between the local gradients $\hat{g}_k^t$ and $\hat{g}_c^t$. Another insight remains that Byzantine attackers usually replace some local gradients with malicious gradients, which directly increases the Euclidean distance between these gradients and the honest gradients. As a result, when the proportion of malicious clients is within a tolerable range, the score of the malicious training clients is expected to be lower than the honest training clients. However, in a scenario without Byzantine attacks, the score represents the degree of heterogeneity of clients. A higher score means a higher degree of heterogeneity.

4.2.2 *Selection Strategy*

We design the selection strategy for determining which uploaded gradient is used to update the global model. Based on our scoring system, we can achieve a secure aggregation by accepting the gradients with high scores, since the malicious gradients obtain low scores. Besides, the convergence analysis and experimental result show that the opposite strategy performs better in a non-attack scenario. Therefore, two opposite selection strategies are designed for different considerations as follows:

- **Selection Strategy I.** The selection strategy I selects several local gradients with relatively higher scores to construct a global gradient for the update of the global model. In other words, we hope that the local gradients which are similar to the committee gradients in the Euclidean space will participate in the construction of the global gradient. In practice, we sort the local gradients according to their scores and only accept the top $\alpha\%$ of them. We design the selection strategy for the consideration of robustness.

Those malicious gradients and honest gradients are far from each other in the Euclidean space, and choosing the gradients close to the committee gradients under the condition that the committee clients are honest can achieve a more robust aggregation process. However, it is difficult for those clients with obvious differences to be selected to participate in the aggregation procedure, which makes it hard for the global model to learn the comprehensive data characteristic, resulting in a decline in performance. Overall, selection strategy I is suitable for FL scenarios with Byzantine attacks.

- **Selection Strategy II.** The selection strategy II selects several local gradients with relatively lower scores to construct a global gradient for the update of the global model. That is, we hope that the local gradients which are different from the committee gradients in the Euclidean space will participate in the construction of the global gradient. Similar to the selection strategy I, we sort the local gradients according to their scores and only accept the bottom $\alpha\%$ of them in practice. We design the selection strategy II for the consideration of convergence rate and performance of the global model. Constructing a global gradient by selecting those local gradients with obvious differences allows the global model to learn more comprehensively, gaining a faster convergence and better performance of the global model. However, this strategy can not provide robustness guarantees, mainly since that the malicious gradients will be preferentially used in global learning, leading to a sharp drop in the performance of the global model. Overall, selection strategy II is suitable for FL scenarios without Byzantine attacks.

4.2.3 Election Strategy

The election strategy is designed for guaranteeing the honesty of committee clients. The committee members reach a consensus on their decisions, and the committee's decision depends on the majority of the committee members. However, the malicious clients mixed into the committee may interfere with the committee's decision-making. Therefore, the committee must guarantee that its honest members are more than malicious members. Otherwise, the committee can not filter out the malicious gradients. We sort the training clients according to their scores and then select these training clients closed to the middle position as the committee clients for the next round. We design the election mechanism based on the following two considerations:

- *Robustness*. According to the above analysis, as malicious training clients will get lower scores than honest training clients, they are expected to locate at the end of the sorted sequence. Therefore, choosing the training client closed to the middle or upper position prevents the malicious training clients from becoming the committee clients in the next round, thereby guaranteeing the security of the global model. Although there may still be a small number of malicious clients mixed into the committee members when the proportion of malicious clients is relatively large, it is difficult to affect the judgment of the whole committee because of their exceeding small proportion of the committee members.
- *System Stability*. Those training clients with the highest scores are not chosen to be the new committee members in order to avoid the system from relying too much on the initialization of committee members. When we choose those training clients with the highest scores to form a new committee, the learning direction of the global model will be completely determined by the initial committee members. It is because those clients with a large Euclidean distance between the local gradient and the committee gradients are not only difficult to be selected to participate in the aggregation procedure, but also lost the opportunity to run for the next round of committee members. This is in line with our intuition that "committee members should be those who can represent the majority".

4.2.4 Committee Consensus Protocol

In our decentralized framework, the committee clients must reach a consensus to complete the scoring, aggregation, selection, and election process. Thus we design a committee consensus protocol, which is inspired by the pBFT [39]. The designed committee consensus protocol(CCP) is as follows(at round t):

1) After the scoring process, each committee client obtains the scores of the training clients. Then every committee client broadcast its scores to the other committee clients. Each committee client is able to calculate the total score of every training client by Eq. (8).
2) A committee client p is selected as the primary committee client by random, while other committee clients are regarded as the replicate committee clients.
3) Each committee client decides its aggregation set $S_{a,c}^t$ according to the selection strategy. Since all scores are broadcasted among committee clients, the aggregation sets among honest committee clients are the same.
4) The primary committee client creates a request $\langle$ *Request*, $S_{a,p}^t$, *operation*, *timestamp* $\rangle$ to ask whether its $S_{a,p}^t$ is correct. Then the primary committee client broadcasts the request to all the replicate committee clients. The operation in the request is to aggregate the local gradients uploaded by aggregation clients as Eq.(5)
5) All the replicate committee clients execute the request. Each of them checks whether the $S_{a,p}^t$ is the same as its own $S_{a,c}^t$. If so, it performs the aggregation process as Eq. (5). After aggregation, it checks whether the result is consistent with the request. If so, the executing result $\langle$ *Reply*, *timestamp*, $S_{a,p}^t$, *response* $\rangle$ is returned to the primary committee client.
6) The primary committee client checks whether it has received at least $\lfloor C/2 \rfloor + 1$ identical results from replicate committee clients. If so, the consensus is reached; otherwise, the primary committee client should be reassigned and steps 3)-6) should be repeated.
7) Similarly, steps 3)-6) are repeated to reach consensus on new committee client set S_c^{t+1}(However, the aggregation client set S_a^t in step 3)-6) should be replaced by new committee client set S_c^{t+1}).

8) The primary committee client broadcasts the global model to all other clients. The next training round is ready to start.

Algorithm 1 Committee Consensus Protocol

```
Input: $t$, $S_c^t$, $S_b^t$
Output: global model $\mathbf{w}^t$, new committee client set $S_c^{t+1}$
1: for $c \in S_c^t$ do
2:   for $k \in S_b^t$ do
3:     The $c$-th committee client broadcasts score $\mathcal{P}_k^c$ to other committee clients.
4:   end for
5: end for
6: for $c \in S_c^t$ do
7:   for $k \in S_b^t$ do
8:     The $c$-th committee client calculates the total score $\mathcal{P}_k$ of the $k$-th training client.
9:   end for
10: end for
11: The committee performs voting($S_a^t$, aggregtion) as Algorithm 2 and gets the global model $\mathbf{w}^t$.
12: The committee performs voting($S_c^{t+1}$, None) as Algorithm 2.
```

Algorithm 2 The voting algorithm

```
Input: The client set $S$ and the opertation $o$
Output: The result $r$ of the operation $o$
1: while No consensus on $S$ do
2:   The primary committee client $p$ is selected from committee clients by random.
3:   for $k \in S_c^t$ do
4:     The $k$-th committee client decides its client set $S^k$ according to the corresponding strategy (selection strategy/election strategy).
5:   end for
6:   $p$ performs the operation $o$ and gets $r$.
7:   $p$ creates a request $\langle$ Request, $S^p$, $o$, timestamp $\rangle$ and broadcasts it to all the replicate committee clients.
8:   for $k \in S_c^t \setminus \{p\}$ do
9:     The $k$-th replicate committee client receives the request from $p$.
10:    if $S^k == S^p$ then
11:      The $k$-th committee client performs the operation $o$.
12:      Return $\langle$ Reply, timestamp, $S^p$, response $\rangle$ to $p$.
13:    end if
14:    if $p$ receives at least $\lfloor C/2 \rfloor + 1$ response then
15:      Consensus is reached on $S$.
16:    end if
17:  end for
18:  Return $r$.
19: end while
```

4.3 Training Algorithm

In this section, we introduce the serverless training algorithm of CMFL. Firstly, all clients randomly initialize the local model and some clients are randomly selected as the committee clients. In each communication round, the algorithm performs the following five steps:

1) *Random Sampling*: A part of the clients from the non-committee clients will be selected randomly as the training clients while the other clients become the idle clients. At round t all training clients and committee clients download the global model from the primary committee client as the local model.
2) *Local Training*: All training clients and committee clients perform τ rounds of SGD over the local datasets and compute the local gradients. The training clients send their local gradients to each committee client for verification.
3) *Gradient Verification*: Each committee client assigns a score on each training client according to the scoring system. Then they execute CCP to reach a consensus on aggregation client set S_a^t.
4) *Global Model Updating*: In CCP, the local gradients uploaded by clients in S_a^t are aggregated for constructing the global gradient, which is used to update the global model according to the Eq. (6) when the consensus is reached.
5) *Election of New Committee Members*: The committee clients execute CCP to reach a consensus on new committee clients set S_c^{t+1}

The algorithm repeats the above five steps until the algorithm converges or t exceeds the defined maximum communication round T.

Algorithm 3 The training algorithm of CMFL

```
Input: $\tau, T, K, m, C, \eta_t, \eta_i^t$
Output: target global model $\mathbf{w}^T$
1: Initialize $\mathbf{w}^1$, $S_c^1$ and $S_b^1$ randomly.
2: for $t = 1$ to $T$ do
3:   for $k \in S_b^t \cup S_c^t$ do
4:     for $i = 1$ to $\tau$ do
5:       The $k$-th client runs the SGD over the local dataset by $\mathbf{w}_{k,i}^t \Leftarrow \mathbf{w}_{k,i-1}^t - \eta_i^t g_k(\mathbf{w}_{k,i-1}^t, \mathcal{B}_{k,i-1}^t)$.
6:     end for
7:   end for
8:   for $k \in S_b^t$ do
9:     for $c \in S_c^t$ do
10:      The $k$-th training client send its local gradient $\hat{g}_k^t$ to the $c$-th committee client.
11:      The $c$-th committee client assign a score $\mathcal{P}_k^c$ on the $k$-th training client/local gradient according to the scoring system.
12:    end for
13:  end for
14:  The committee clients execute CCP to complete the selection, aggregation, and election process.
15:  The $S_b^t$ be reinitialized to form the $S_b^{t+1}$.
16:  for $k \in S_b^{t+1} \cup S_c^{t+1}$ do
17:    The $k$-th client downloads the global model from the primary committee client.
18:  end for
19: end for
```

5 Theoretical Analysis

In this section, we show the theoretical analysis of CMFL, including convergence analysis and complexity analysis.

5.1 Convergence Analysis

In this section, we conduct the convergence analysis of the proposed framework CMFL. First, we introduce some basic assumptions used for the convergence analysis in Section 5.1.1. Second, we introduce two definitions that facilitate our analysis in Section 5.1.2. Finally, we present our result and analysis for convergence of CMFL in Section 5.1.3. The proof of the convergence is presented in the Supplement.

For the purpose of convergence proof and analysis, we define $F^* = \min_{\mathbf{w}} F(\mathbf{w}) = F(\mathbf{w}^*)$ as the optimal value of the global objective function, where $\mathbf{w}^*$ denotes the optimal global model. In the same way we define $F_k^* = \min_{\mathbf{w}} F_k(\mathbf{w}) = F_k(\mathbf{w}_k^*)$ as the optimal value of the k-th client's local objective function, where $\mathbf{w}_k^*$ denotes the optimal local model of k-th client.

5.1.1 Assumptions

First, we introduce the assumptions used for convergence analysis.

Assumption 1. (Lipschitz Gradient). *$F_1, ..., F_K$ are all $\mathcal{L}$-smooh: for all $\mathbf{v}, \mathbf{w} \in R^n$, $k = 1, ..., K$, $F_k(\mathbf{v}) \leq F_k(\mathbf{w}) + (\mathbf{v}-\mathbf{w})^T \nabla F_k(\mathbf{w}) + \frac{L}{2}||\mathbf{v}-\mathbf{w}||_2^2$.*

Assumption 2. (μ-strongly Convex Gradient). *$F_1, ..., F_K$ are all μ-strongly convex: for all $\mathbf{v}, \mathbf{w} \in R^n$, $k = 1, ..., K$, $F_k(\mathbf{v}) \geq F_k(\mathbf{w}) + (\mathbf{v}-\mathbf{w})^T \nabla F_k(\mathbf{w}) + \frac{\mu}{2}||\mathbf{v}-\mathbf{w}||_2^2$*

Assumption 3. (Bounded Variance). *For the mini-batch $B_{k,i}^t$ uniformly sampled randomly from k-th client's dataset $\mathcal{D}_k$, the resulting stochastic gradient is unbiased: $E[g_k(\mathbf{w}_{k,i}^t, \mathcal{B}_{k,i}^t)] = \nabla F_k(\mathbf{w}_{k,i}^t)$ for all $k = 1, ..., K, t = 1, ..., T, i = 1, ..., \tau$. And the variance of stochastic grandient in each client is bounded: $E||g_k(\mathbf{w}_{k,i}^t, \mathcal{B}_{k,i}^t) - \nabla F_k(\mathbf{w}_{k,i}^t)||^2 \leq \sigma^2$.*

Assumption 4. (Bounded Gradient). *The local gradient's expected squared norm is uniformly bounded: $\mathbb{E}||g_k(\mathbf{w}_{k,i}^t, \mathcal{B}_{k,i}^t)||^2 \leq G^2$ for all $k = 1, ..., K, t = 1, ..., T, i = 1, ..., \tau$.*

Assumption 5. (Bounded Objective Function). *For any aggregation client set $S_a \notin \varnothing$ and the optimal committee client set $S_c^* \notin \varnothing$, the difference of local optimal objective function is bounded: $\mathbb{E}[||\sum_{k\in S_a} p_{k,S_a} F_k^* - \sum_{k'\in S_c^*} p_{k',S_c^*} F_{k'}^*||] \leq \kappa^2$, where S_c^* satisfies that $S_c^* = \arg\min_{S_c} \sum_{k\in S_c} p_{k,S_c} F_k^*$ and $|S_c^*| = C$.*

Assumption 1 and 2 are standard conditions in FL setting [40] [41] [42] [43] and many common machine learning optimization algorithms meet these assumptions, such as the ℓ_2-norm regularized linear regression, logistic regression, and softmax classifier [28]. Assumption 3 is a form of bounded variance between the local objective functions and the global objective function [44], and Assumption 4 is fairly standard in nonconvex optimization literature [45] [46] [47] [48]. They are widely used in the FL convergence analysis, such as Li *et al.* [28] and Cho *et al.* [49]. Assumption 5 is used to constrain the optimal objective function deviation between the aggregation client set and the committee client set caused by the committee mechanism. For the need of convergence proof, we define S_c^* as the set which contains C clients with the smallest optimal local objective function F_k^*.

5.1.2 Definition

We introduce two related definitions for the convenience of analysis as follows.

Definition 1. **(Degree of Heterogeneity).** *We use*

$$\Gamma = F^* - \sum_{k=1}^{K} p_k F_k^* = \sum_{k=1}^{K} p_k (F_k(\mathbf{w}^*) - F_k(\mathbf{w}_k^*)) \quad (9)$$

to quantify the degree of heterogeneity among the clients.

Li *et al.* [28] proposed this definition, which is widely used in the convergence analysis of FL on Non-IID dataset [50]. In the Non-IID FL scenarios, the Γ remains nonzero and its value reflects the heterogeneity of the data distribution. In the IID FL scenarios, with the growth of K the Γ gradually goes to zero.

Definition 2. **(Aggregation-Committee Gap).** *For any aggregation client set S_a^t, we define*

$$\begin{aligned}&\varphi(S_a^t, \mathbf{w}) \\ =&\frac{\mathbb{E}[\sum_{k\in S_a^t} p_{k,S_a^t} F_k(\mathbf{w}) - \sum_{k'\in S_c^*} p_{k',S_c^*} F_{k'}^*]}{F(\mathbf{w}) - \sum_{k=1}^{K} p_k F_k^*} \geq 0,\end{aligned} \quad (10)$$

where $\mathbb{E}$ denotes the expectation over all randomness in the previous iterations, and S_c^* denotes the optimal committee client set.

$\varphi(S_a^t, \mathbf{w})$ changes with the changes of S_a^t and $\mathbf{w}$ during training. An upper bound φ_{max} and a lower bound φ_{min} are defined to obtain a conservative error bound in the convergence analysis:

$$\varphi_{min} = \min_{S_a^t, \mathbf{w}} \varphi(S_a^t, \mathbf{w}), \varphi_{max} = \max_{S_a^t} \varphi(S_a^t, \mathbf{w}^*). \quad (11)$$

5.1.3 Convergence Result and Analysis

We analyze the convergence of Algorithm 3 in this section and find an upper bound of $\mathbb{E}[F(\overline{\mathbf{w}}^{(T)})] - F^*$, which denotes the convergence error of the global model after T rounds:

Theorem 1. (Convergence of Committee Mechanism based Federated Learning). *Under Assumption 1 to 5, Definition 1 to 2 and the learning rate η_t, where $\eta_t = \frac{1}{\mu(t+\gamma)}$ and $\gamma = \frac{4L}{\mu}$, the error after T rounds of CMFL satisfies*

$$\begin{aligned}&\mathbb{E}[F(\overline{\mathbf{w}}^T)] - F^* \\ \leq&\frac{1}{T+\gamma}\left[\frac{4L(32\tau^2G^2 + \sum_{k=1}^{K} p_k\sigma_k^2) + 24L^2\kappa^2}{3\mu^2\varphi_{min}}\right. \\ &\left.+\frac{8L^2\Gamma}{\mu^2} + \frac{L\gamma||\overline{\mathbf{w}}^1 - \mathbf{w}^*||^2}{2}\right] + \frac{8L\Gamma}{3\mu}\left(\frac{\varphi_{max}}{\varphi_{min}} - 1\right),\end{aligned} \quad (12)$$

where L, μ and σ represent the constant light indicated in the Assumption 1, 2 and 3. G and κ represent the upper bound values defined in Assumption 4 and 5 .Γ denotes the heterogeneity among the clients according to the Definition 1. These are all constant while φ_{min} and φ_{max} will be different depending on the selection strategy. The impact of selection strategy on φ_{min} is analyzed below. CMFL affects the performance of the global model by altering φ_{min}. The proof of the theorem is shown in Supplement.

The impact of selection strategy on φ_{min}. According to the definition, the value of φ is positively correlated with $\sum_{k\in S_a^t} p_{k,S_a^t} F_k(\mathbf{w})$, which represents the average local loss of the model over the aggregation client set S_a^t. Under our framework, the lower bound of $\varphi(S_a^t, \overline{\mathbf{w}}^t)$ defined as φ_{min} affects the convergence rate of the global model, where $\overline{\mathbf{w}}^t$ represents the global model at round t. In general, those clients whose data set distribution is similar to that of the majority have a relatively low local loss, while others have a relatively high local loss. We call the former *universal clients* and the latter *extreme clients*. The tendency to choose a universal client for aggregation results in low φ_{min}, and the tendency to choose an extreme client for aggregation results in high φ_{min}. The building of the election strategy ensures that committee members can represent the majority, and the scoring system is designed as the clients similar to committee members can get higher scores, so clients with high scores are more likely to have a low local loss. When we adopt selection strategy I, we get a low φ_{min}. Instead, when we adopt selection strategy II, we get a high φ_{min}.

Effect of φ_{min} on convergence rate. Note that a higher φ_{min} results in faster convergence at the rate $\mathcal{O}(\frac{1}{T\varphi_{min}})$. That is, adopting the selection strategy I make the convergence of the global model slows down, while the selection strategy II accelerates the convergence of the global model, which is verified in the following experiments. However, considering the reality of Byzantine attacks, the first selection strategy is a more appropriate choice.

5.2 Complexity Analysis

In this section, we analyze the computation complexity and communication complexity of the proposed framework CMFL. For each complexity analysis, the time and overhead of CMFL are considered. Recall the previous notations, we define C as the number of committee clients, n as the training clients. Noted that m represents the number of aggregation clients.

5.2.1 Computaion Complexity

There are five phases related to computation complexity as follows:

- **Local Training**. Each training client and committee client performs SGD locally before they upload their local gradients. The computation overhead of k-th client is $\mathcal{O}(|\mathcal{D}_k| \cdot |\mathbf{w}|)$, where $|\mathcal{D}_k|$ is the number of data samples, $|\mathbf{w}|$ is the model size. Assumed that $|\mathcal{D}|$ represents the average value of $|\mathcal{D}_k|$ over all clients, the total computation overhead is $\mathcal{O}((n + C)|\mathcal{D}| \cdot |\mathbf{w}|)$. Because each client performs local training in parellel, the computation time of local training is $\mathcal{O}(|\mathcal{D}| \cdot |\mathbf{w}|)$.
- **Scoring**. In the phase of scoring, each committee client assigns a score to each training client. According to the scoring system, the computation overhead of scoring for one committee client is $\mathcal{O}(n \cdot |\mathbf{w}|^2)$. The total computation overhead of scoring is $\mathcal{O}(n \cdot C \cdot |\mathbf{w}|^2)$. As each committee client performs scoring operation in parallel, the computation time of scoring is $\mathcal{O}(n \cdot |\mathbf{w}|^2)$.
- **Aggregation**. In the phase of aggregation, only m local gradients are used for aggregation, so the computation overhead of aggregation for one committee client is $\mathcal{O}(m \cdot |\mathbf{w}|)$. According to the CCP, each committee client performs aggregation so the total computation overhead of aggregation is $\mathcal{O}(m \cdot C \cdot |\mathbf{w}|)$. Since the aggregation process is performed in parallel, the computation time of aggregation is $\mathcal{O}(m \cdot |\mathbf{w}|)$.
- **Selection**. In the phase of selection, each committee client sorts the received local gradients and selects m local gradients for aggregation. The computation overhead is $\mathcal{O}(nlogn)$, which can be ignored since it is much smaller than the computation overhead of the above three phases.
- **Election**. In the phase of the election, each committee client determines its new committee client set based on the sorted local gradients. The computation overhead is $\mathcal{O}(1)$, which can be ignored because it is much smaller than the computation overhead of the above four phases.

5.2.2 Communication Complexity

Assumed that each client owns the same maximum bandwidth, and r represents the max transmission rate, the communication time of transferring data s is computed as $T_{transmission} = s/r$. There are three phases related to communication complexity as follows:

- **Gradient Uploading**. After local training each training client uploads its local gradient to all the committee clients, so the communication overhead of uploading for one client is $\mathcal{O}(C \cdot |\mathbf{w}|)$. The total communication overhead of uploading is $\mathcal{O}(n \cdot C \cdot |\mathbf{w}|)$. Since each training client performs the uploading process in parellel, the communication time of uploading is $\mathcal{O}(C \cdot |\mathbf{w}|/r)$.
- **Global Model Downloading**. After global aggregation, the primary committee client broadcasts the global model to all training clients in the next round. The total communication overhead of downloading is $\mathcal{O}(n \cdot |\mathbf{w}|)$, thus the communication time of downloading is $\mathcal{O}(n \cdot |\mathbf{w}|/r)$.
- **Broadcasting**. Committee clients should perform CCP to reach consensus, in this phase the primary committee client broadcast S_a^t and S_c^{t+1} to other committee clients, which occur communication overhead. However, this communication overhead can be ignored because it is much smaller than the communication overhead of uploading and downloading.

Besides, system heterogeneity is a widespread problem in the field of Federated Learning. The differences in the computational performance of clients lead to various time consumptions, which becomes the bottleneck limiting the efficiency of the Federated Learning system. Some clients even drop out during the training process. There are some methods to alleviate the performance degradation caused by system heterogeneity, such as setting a maximum waiting time or a minimum number of received gradients. Indeed, system heterogeneity is an important issue in Federated

Learning, and more works need to be carried out to address it.

6 Experiments

In this section, we first present our experimental setup in Section 6.1, which includes the datasets, models, and experimental environment. Then, we evaluate our proposed framework CMFL by five sets of following experiments, and the results and analysis are presented in Section 6.2 to Section 6.6.

1) **Normal Training Experiment.** We test the performance of CMFL without considering the Byzantine attack and compare it with typical FL.
2) **Robustness Comparative Experiment.** We evaluate the Byzantine resilience of CMFL and compare it with several Byzantine-tolerant algorithms.
3) **Hyperparameter Analysis Experiment.** We vary the hyper-parameter α, ω, and ϵ and show how it affects the performance.
4) **Efficiency Experiment.** We evaluate the efficiency of CMFL and compare it with other decentralized FL frameworks while considering the computation and communication overhead.
5) **Committee Member Analysis Experiment.** We track the malicious clients that have been mixed into the committee members to analyze the impact of these malicious clients on the performance of the global model.

6.1 Experimental Setup

6.1.1 Datasets and Models

We evaluate CMFL over three datasets with a Non-IID setting, including FEMNIST, Sentiment140, and Shakespeare.

- **FEMNIST-AlexNet.** [51] FEMNIST (Federated Extended MNIST) is a real-world distributed dataset formed by a specific division of the EMNIST dataset. This dataset is used to train a model for handwritten digit/character recognition tasks, which contains 805263 images of 28×28 pixels, divided into 64 categories, each category represents a type of handwritten digit/character ($0-9$ and $a-z$). The FEMNIST dataset divides the EMNIST dataset into 3550 parts in a specific way and stores them on each client to simulate a real federated learning scenario. We use the convolutional neural network AlexNet as the basic experimental model for this image classification dataset.
- **Sentiment140-LSTM.** [52] Sentiment140 is a real-world distributed dataset which focus on the text sentiment analysis task, including 1,600,000 tweets extracted using the Twitter API. The data in this dataset has been annotated (0 = negative, 2 = neutral, 4 = positive). It can be used to discover the sentiment of a brand, product, or topic on Twitter. We regard each Twitter account as a client and choose a two-layer LSTM binary classifier as the basic experimental model. The LSTM binary classifier includes 256 hidden units with pretrained 300D GloVe embedding [53].
- **Shakespeare-LSTM.** [54] Shakespeare is a real-world distributed dataset built from *The Complete Works of William Shakespeare*. Each speaking role corresponds to a different device, and we choose a two-layer LSTM classifier as the basic experimental model, which contains 100 hidden units with an 8D embedding layer. There are 80 classes of characters in dataset. The model takes a sequence of 80 characters as input, embeds each character into a learned 8-dimensional space, and outputs one character for each training sample after 2 LSTM layers and a densely connected layer.

The Non-IID levels of these two datasets are shown in Table 2.

6.1.2 Environment

Global Setup: At the beginning of each communication round, 10% of the idle clients are activated to participate in the training, while the rest clients wait for the next selection. At one communication round, each client runs one iteration of SGD for local training. Besides, the initial committee clients are selected from all clients at random. In order to better demonstrate the experimental effect, we set different learning rates η in different datasets.

Machines: We perform our experiment on a commodity machine with Intel Core CPU i9-9900X containing a clock rate of 3.50 GHz with 10 cores and 2 threads per core. And we utilize Geforce RTX 2080Ti GPUs to accelerate training. The learning model is implemented in Python 3.7.6 and Tensorflow 1.14.

Hyper-parameter Notation: In each communication round, 10% of the whole clients are activated to participate in the training of that round, ω% of which become the committee clients and the rest become the training clients. Each round α% of the training clients become the aggregation clients, whose local gradients are accepted to use for the constructing of the global gradient. ϵ presents the percentage of malicious clients.

6.2 Nomal Training Experiment

6.2.1 Experiment Setting

In this experiment we set $\alpha = 40$, $\omega = 40$ and $\epsilon = 0$ to simulate a non-attack FL scene. The η in FEMNIST and Sentiment140 is 0.001 and 0.005 respectively. In this experiment, we compared typical FL and CMFL under these two selection strategies:

- **Typical FL.** Typical FL uses all the local gradients uploaded by the training client for the construction of global gradients.
- **CMFL with Selection Strategy I.** The original intention of this selection strategy is to resist Byzantine attacks. The committee accepts the local gradients uploaded by the high-score training clients while rejects those uploaded by low-score training clients.
- **CMFL with Selection Strategy II.** This selection strategy is opposite to selection strategy I, accepting the local gradients uploaded by the low-score clients while rejects those uploaded by high-score training clients.

TABLE 2: Statistics of datasets

Dataset	Number of devices	Total samples	Mean	Stdev
FEMNIST	3,550	805,263	226.83	88.94
Sentiment140	772	40,783	53	32
Shakespeare	143	517,106	3616	6808

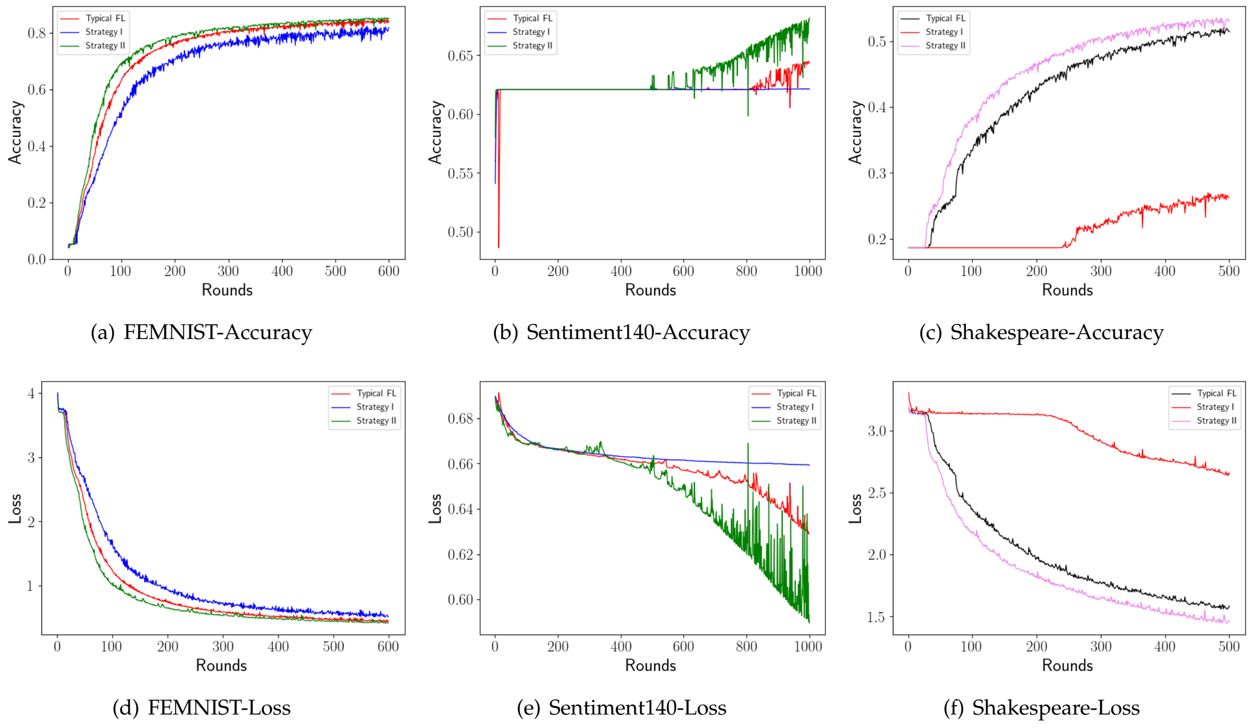


Fig. 2: The performance of global model among the typical FL and CMFL with two selection strategies.

6.2.2 *Result Analysis*

We show the result in Figure 2. Note that the performance of CMFL under selection strategy II is better than that of CMFL under selection strategy I and typical FL. The CMFL under the selection strategy I has the slowest model convergence rate and the worst model accuracy among the three. The result shows that it is a more appropriate design for the committee to accept the local gradients uploaded by the low-score clients when in non-attack FL scene, which can not only enhance the global model performance but also accelerate the convergence of the global model. This is because over the Non-IID dataset, the CMFL under the selection strategy I makes it difficult for a few clients which are quite different from other clients to participate in the aggregation process, resulting in the training of the global model only using part of the data instead of all of the data. Figure 3(a) proves it. We record the aggregation times of each client and plot them as a curve graph. From the curve, we can see that the aggregation times of client under FL conform to a Gaussian distribution, causing FL selects the aggregation randomly. Besides, we found that a high percentage of clients never participate in the aggregation process when performing CMFL under selection strategy I. That is why CMFL under selection strategy I achieves worse performance than typical FL and CMFL under selection strategy II. It is difficult for the global model to learn comprehensive knowledge while using this kind of training algorithm. However, the CMFL under the selection strategy II significantly reduces the number of such clients. In this way more clients can participate in the aggregation process, making the global model learn a more comprehensive knowledge. Figure 3(b) and 3(c) explain why CMFL under selection strategy II achieves a better performance than typical FL. We record the testing accuracy of each client after training. Figure 3(b) shows that CMFL under selection strategy II has least clients with low accuracy, while has most clients with high accuracy. That is because those clients with low accuracy have more opportunities to participate in the aggregation process, which is proved by Figure 3(c). Figure 3(c) shows that the aggregation times of clients with low accuracy in CMFL under selection strategy II are much more than other training methods. By reducing the proportion of clients with low accuracy, CMFL under selection strategy II achieves a better performance than typical FL.

Nevertheless, it is not advisable to choose strategy II when considering the Byzantine attack. According to the scoring system we designed, the malicious client will get a lower score than the honest client. Taking the selection strategy II means that Byzantine attackers can easily attack the global model by uploading malicious local gradients. In this scenario, choosing strategy I is a more appropriate choice by avoiding the local gradients with low scores to participate in the aggregation, which is likely to be malicious gradients. Although the effect of CMFL under strategy I in the non-attack FL scenario is slightly weaker than that of the typical FL, it can make the training process more robust in a wider range of realistic scenarios.

6.3 Robustness Comparative Experiment

6.3.1 Experiment Setting

In this experiment we set $\alpha = 40$, $\omega = 40$ and $\epsilon = 10$, and we test the Byzantine resilience of CMFL under two selection strategies by comparing it with the following Byzantine-tolerant algorithms, which are all aim to design a robust gradient aggregation method.

- **Median** [12]: This aggregation method first sorts the gradients and selects the median to be the global gradient.
- **Trimmed Mean** [12]: This aggregation method first sorts the local gradients, then removes the maximum value of $\beta\%$ and the minimum value of $\beta\%$, and finally aggregates the remaining local gradients to construct the global gradient, where $\beta \in [0, 50)$.
- **Krum** [4]: This aggregation method assumes that the directions of the local gradients uploaded are relatively similar and sort these local gradients according to their similarity. The original Krum algorithm only selects the first local gradient after sorting as the global gradient.
- **Multi-Krum** [4]: The Multi-Krum algorithm is an improved version of the original Krum, which selects the top $\alpha\%$ of the sorted local gradients to construct the global gradient. Compared with the original Krum, Multi-Krum reduces the fluctuation of the global model effect caused by random factors, making the training process more stable. Since Multi-Krum uses more local gradients to construct the global gradients, it can make the global model converge faster than the original Krum.

We compare these Byzantine-tolerant algorithms with CMFL considering three different types of attacks: gradient scaling attack, same-value gaussian attack, and back-gradient attack, where the attackers are all aim to compromise some clients and upload the malicious gradients.

- **Gradient Scaling Attack**: The malicious clients multiply each element in the local gradient by a random value $\lambda \in [a, 1)$, where a is a defined constant which indicates the magnitude of the attack. In this experiment we set $a = 0.5$.
- **Same-value Attack** [55]: The malicious clients replace the local gradient with a vector of the same size whose elements are all 0.
- **Back-gradient Attack** [56]: The malicious clients replace the local gradient with a vector of the same size in the opposite direction.

6.3.2 Result Analysis

We show the performance of CMFL compared with other Byzantine-tolerant algorithms under various attacks in Figure 4-6. We analyzed the performance of each Byzantine-tolerant algorithm.

- **CMFL-selection strategy I** always reaches the fastest global convergence speed and highest accuracy among these five algorithms. Note that in the model accuracy curve, the CMFL curve fluctuates more obviously than Median and Multi-Krum, which is the normal curve fluctuation caused by the replacement of committee members. Nevertheless, its overall accuracy rate is still higher than the other algorithms.
- **CMFL-selection strategy II** achieves as poor performance as typical FL. Obviously, the malicious clients can easily participate in the aggregation process.
- **Median** [12] has maintained a relatively stable performance under a variety of attacks. It only selects the median of the local gradients as the global gradient, which is regarded as a relatively conservative approach. Although Median can effectively resist Byzantine attacks, it is difficult to achieve excellent model performance.
- **Trimmed Mean** [12] has an unstable performance under different attacks. Trimmed Mean performs well under gradient scaling attack but still not as good as Median and Multi-Krum, and performed extremely badly under same-value attack and back-gradient attack. This Byzantine-tolerant algorithm cannot effectively resist the Byzantine attacks.
- **Krum** [4] performs poorly under all three attacks, and there were sharp fluctuations in the training process. Krum only selects the first local gradient after sorting as the global gradient each time. This aggregation method that does not consider the overall results in severe fluctuations of the global model.
- **Multi-Krum** [4] has the second-best model performance overall. Multi-Krum improves the original Krum and eliminates the jitter generated during the training process.

Note that the CMFL is the only framework without involving a central server, which naturally achieves robustness against the influence of a malicious server. Other Byzantine-tolerant algorithms cannot achieve the same robustness owing to their naturally centralized architecture.

6.4 Hyper-parameter Analysis Experiment

6.4.1 Experiment Setting

In this experiment, we consider the effect of hyper-parameter by varying the hyper-parameter α, ω and ϵ. Specifically, we consider the back-gradient attack and designed three sets of sub-experiments. In each set of sub-experiments, we fixed one of the hyper-parameters and changed the other two hyper-parameters to analyze their impacts on the model performance.

- **Sub-experiment I.** Fixed $\alpha = 40$ and vary ω, ϵ to $\{10, 20, 30, 40, 50\}$.
- **Sub-experiment II.** Fixed $\omega = 40$ and vary α, ϵ to $\{10, 20, 30, 40, 50\}$.
- **Sub-experiment III.** Fixed $\epsilon = 10$ and vary α, ω to $\{10, 20, 30, 40, 50\}$.

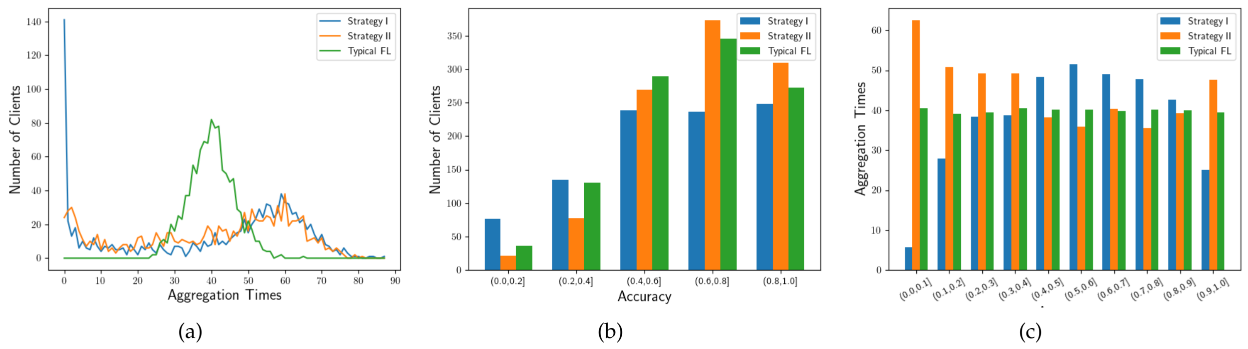


Fig. 3: Figure (a) shows the number of clients with different aggregation times. Figure (b) shows the number of clients with different accuracy. Figure (b) shows the aggregation times of clients with different accuracy.

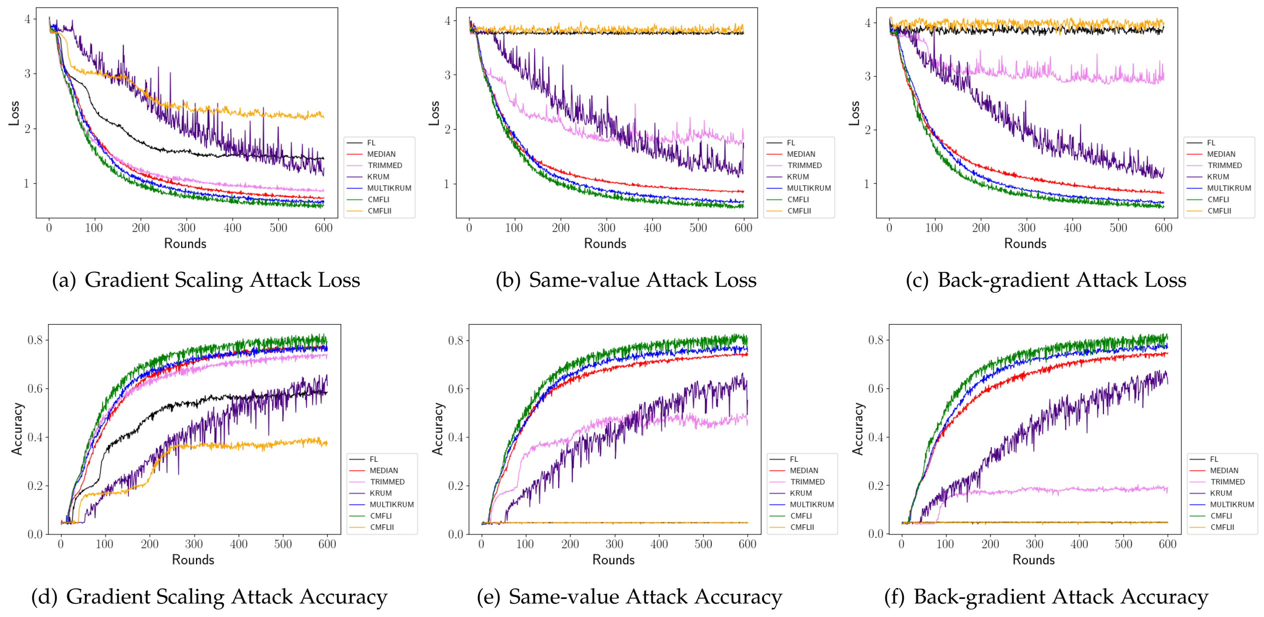
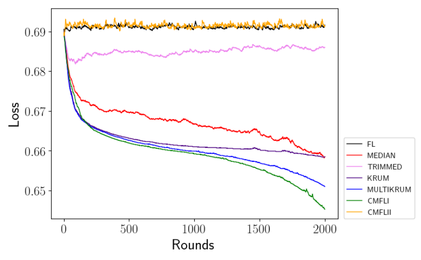


Fig. 4: Performance of CMFL compared with other Byzantine-tolerant algorithms under various attacks over FEMNIST dataset. CMFLI represents CMFL under selection strategy I and CMFL II represents CMFL under selection strategyII.

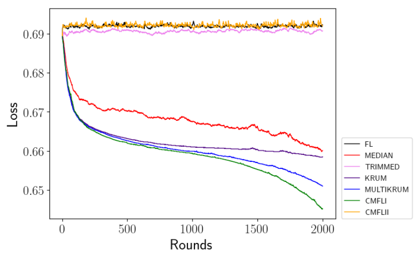
6.4.2 Result Analysis

We show the results in Figure 7 and the analysis as follows:

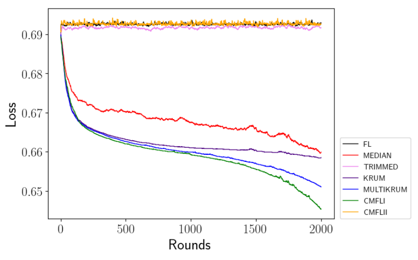
- **Appropriately increasing the number of committee members can enhance the performance of the global model.** The results show that within a suitable parameter value range (e.g., $\alpha = 10, \epsilon \leq 30, \omega \leq 30$), more committee members lead to better global model performance. This is mainly since an appropriate increase in the number of committee members can enhance the robustness of the committee to a certain extent, in the meanwhile avoid the existence of "dictators" and prevent the training process from being controlled by a small number of clients, which makes it difficult for some other clients to participate in the aggregation process.
- **Appropriately increasing the proportion of the aggregation clients can enhance the performance of the global model.** Within a suitable parameter value range, a higher proportion of the aggregation clients leads to better global model performance. Because in the normal operation of the committee, sorting the local gradients uploaded by the training client according to their scores makes the honest local gradients come first and the malicious gradients second. In this ideal situation, we control the proportion of aggregated clients to be less than a threshold χ to prevent malicious gradients from participating in the aggregation process. Under the limit of $\alpha \in (0, \chi]$, the increase of α means that more honest local gradients are selected to construct the global gradient in each round, which can achieve better global model performance.
- **Excessive α, ω, and ϵ will lead to a cliff-like decline in the performance of the global model.** When we increase the proportion of committee members, it means that both malicious clients and honest clients



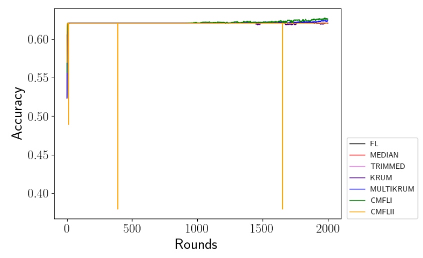
(a) Gradient Scaling Attack Loss



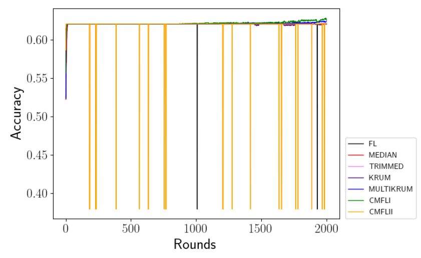
(b) Same-value Attack Loss



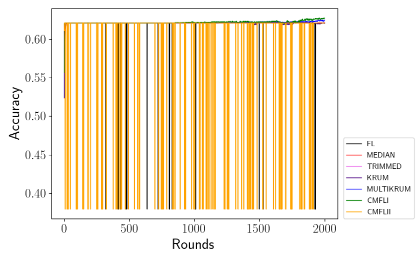
(c) Back-gradient Attack Loss



(d) Gradient Scaling Attack Accuracy

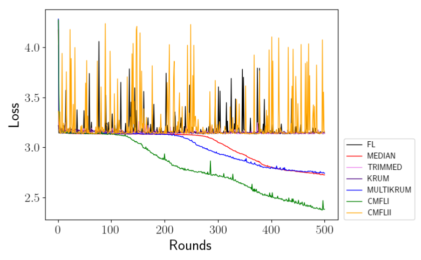


(e) Same-value Attack Accuracy

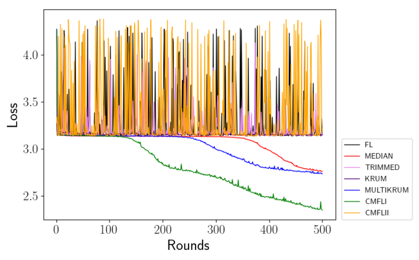


(f) Back-gradient Attack Accuracy

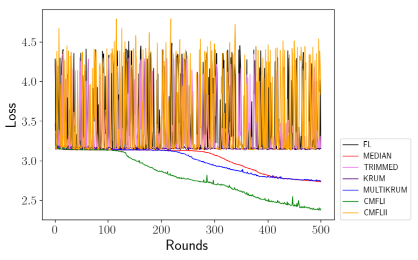
Fig. 5: Performance of CMFL compared with other Byzantine-tolerant algorithms under various attacks over Sentiment140 dataset.



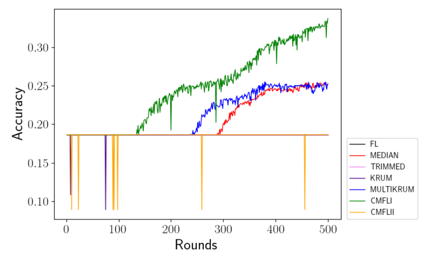
(a) Gradient Scaling Attack Loss



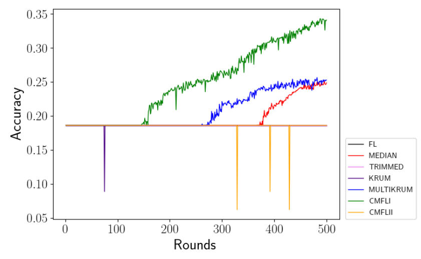
(b) Same-value Attack Loss



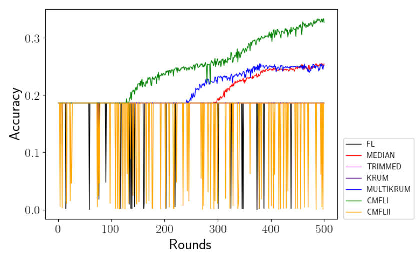
(c) Back-gradient Attack Loss



(d) Gradient Scaling Attack Accuracy

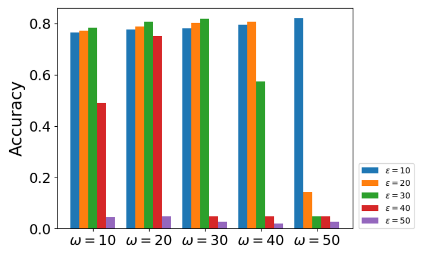


(e) Same-value Attack Accuracy

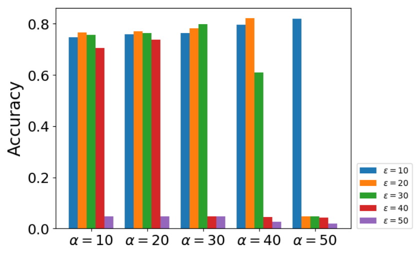


(f) Back-gradient Attack Accuracy

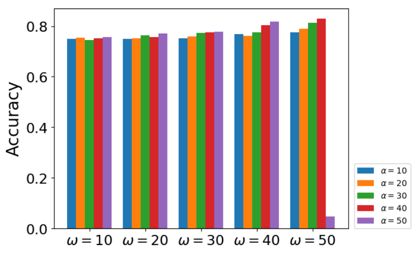
Fig. 6: Performance of CMFL compared with other Byzantine-tolerant algorithms under various attacks over Shakespeare dataset.



(a) $\alpha = 40$



(b) $\omega = 40$



(c) $\epsilon = 10$

Fig. 7: Model performance of CMFL with varying the hyper-parameters.

have a greater probability of being elected as committee members. This increases the probability of malicious clients mixing into committee members and damage the committee's scoring system, making it difficult to assign the correct score to the training clients. In the worst situation, once the proportion of the malicious clients among committee members is relatively large, the scores of the malicious clients will be higher than those of the honest clients, resulting in the malicious local gradient uploaded by the malicious clients being used to construct the global gradient. Hence, the performance of the global model will be devastatingly damaged. When we increase the proportion of the aggregation clients, it also means increasing the probability of malicious local gradients participating in the aggregation procedure. When the proportion of aggregation clients exceeds the threshold that the system can tolerate, the malicious local gradients are used to construct the global gradient, which causes irreparable damage to the performance of the global model. By the same token, if there are too many malicious clients, the malicious local gradients uploaded by them are more likely to participate in the aggregation process. As long as a malicious local gradient is aggregated, the performance of the global model will be greatly reduced.

6.5 Efficiency Experiment

In this section, we evaluate the efficiency of CMFL over FEMNIST and Sentiment140 dataset. Besides, we compare it with other decentralized FL frameworks.

6.5.1 Experiment Settting

Implementation Detail. In this experiment, we simulate the data transferring by calculating the transmission time using $T_{transmission} = s/r$, where s represents the data and r represents the transmission rate. We let the program process wait for $T_{transmission}$ second when it needs to transmit data for simulating the data transferring in a real scene. The maximum number of communication rounds is 600.

Hyper-parameter Setting. In order to exclude the influence of irrelevant variables, we conduct this evaluation without considering the Byzantine attacks. The number of committee clients and training clients is set as 43 and 65. And the aggregation rate α is set as 40%. We set different maximum transmission rates for one client: $1Mps$, $10Mps$, and $100Mps$, and under each transmission rate, we evaluate the performance of CMFL using wall-clock time, which includes computation time and communication time.

Comparative Method. We compare the efficiency of CMFL with three algorithms: typical FL, BrainTorrent [57], and GossipFL [58].

6.5.2 Result Analysis

Figure 8 shows the result. Noted that the size of AlexNet ($\approx$ 200M) is much bigger than that of LSTM ($\approx$ 16k), so the communication time over FEMNIST dataset is much longer than that over the Sentiment140 dataset. The overall performance of CMFL is better than the other two decentralized FL frameworks but worse than typical FL. Typical FL has an overall better performance than CMFL because the clients do not have to communicate with each other. It is a centralized framework so the clients just send their local gradients to a server for aggregation. Never can the other three decentralized frameworks do that because the client has to broadcast their local gradients to other clients for reaching consensus, in which case the transmission of the local gradients occurs communication overhead. With the increasing transmission rate, the advantage of typical FL becomes smaller. The performance of the typical FL is taken over by CMFL when the transmission is up to $100Mps$ over the Sentiment140 dataset. CMFL always has a better performance than the other two decentralized frameworks, because it utilized the committee system to reduce the communication overhead, which is proved in Figure 9. The clients do not need to broadcast their local gradients to other clients but send their local gradients to committee clients. The committee clients can reach a consensus with lower communication overhead by performing CCP, while the other two decentralized frameworks must cost higher communication overhead. Considering the malicious server in a real scenario, CMFL can achieve both robustness and efficiency.

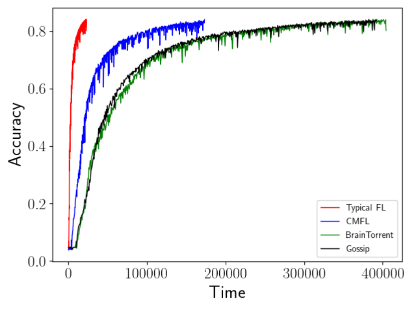
6.6 Committee Member Analysis Experiment

6.6.1 Experiment Setting

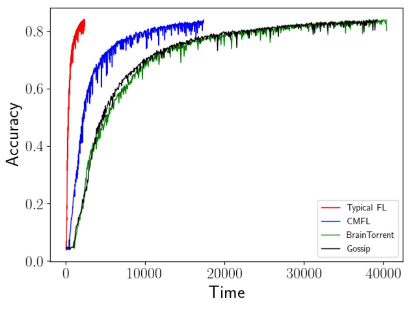
In this experiment we set $\alpha = 40, \omega = 30$ and vary ϵ to $\{10, 20, 30, 40, 50\}$. By recording the number of malicious training clients, malicious committee clients and malicious aggregation clients during the training process, we analyze the influence of committee members on the performance of the global model.

6.6.2 Result Analysis

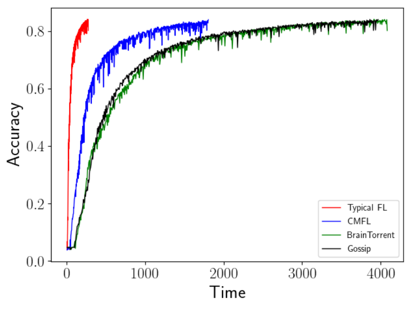
We show the result in Figure 10. Since in each round the training clients are randomly selected from the non-committee clients, when ϵ increases, the number of malicious training clients also increases. And with the increase of ϵ, some malicious clients will inevitably be mixed into the committee members. Nevertheless, a small number of miscellaneous malicious clients cannot destroy the entire committee's scoring system and influence the committee's judgment, which instead improve the performance of the global model. This is mainly because the scoring system does not lose the ability to distinguish between the malicious clients and the honest clients since those malicious clients still receive a score much lower than that of honest clients. But for the honest clients, they get almost the same score due to the extreme scores assigned by the malicious committee clients. In this case, our method is equivalent to the typical FL method without considering the Byzantine attack. The typical FL achieves better performance than CMFL in such a setting, which has shown in Experiment 6.2. However, if there are too many mixed malicious clients, the committee's scoring system will be destroyed and the committee members will lose the ability to eliminate malicious local gradients, resulting in a sharp drop in the performance of the global model.



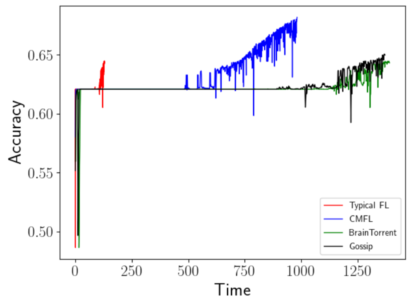
(a) FEMNIST-1Mps



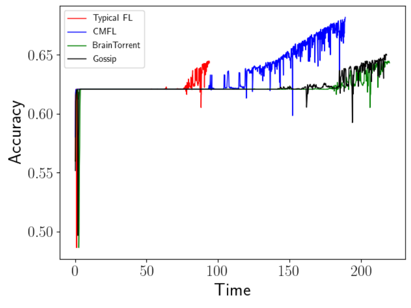
(b) FEMNIST-10Mps



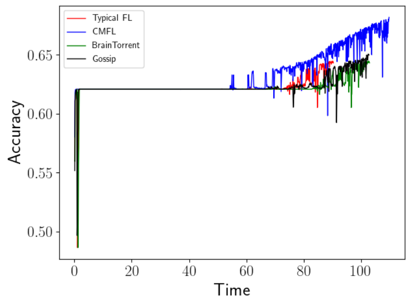
(c) FEMNIST-100Mps



(d) Sentiment140-1Mps

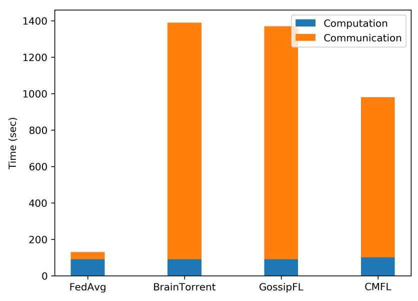


(e) Sentiment140-10Mps

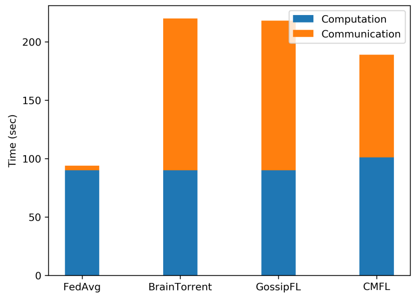


(f) Sentiment140-100Mps

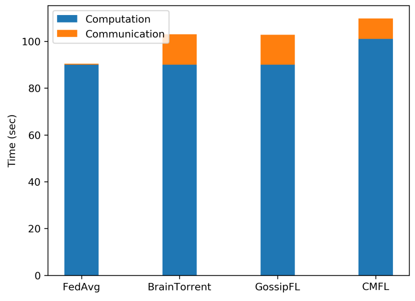
Fig. 8: Performance of typical FL and three decentralized FL frameworks under different transmission rates over FEMNIST dataset and Sentiment140 dataset.



(a) Sentiment140-1Mps



(b) Sentiment140-10Mps



(c) Sentiment140-100Mps

Fig. 9: The total communication time and computation time of typical FL and three decentralized FL frameworks over the Sentiment140 dataset.

7 Conclusion

In this paper, we propose a serverless FL framework under committee mechanism, which can ensure robustness when considering Byzantine attacks. Besides, we present the convergence guarantees for our proposed framework. Motivated by the insight from the theoretical analysis we design the election and selection strategies, which empower the model the robustness against both the Byzantine attack and malicious server problem. The experiment results demonstrate the outperformance of the model over the typical federated learning and Byzantine-tolerant models, which further verify the effectiveness and robustness of the proposed framework.

Currently, the proposed framework mainly ensures the robustness of the aggregation procedure by detecting the abnormal local gradients. While in the face of targeted attacks, such as a backdoor attack, how to design suitable election and selection strategies with theoretical guarantee remains an open and worth-exploring topic in the future.

Acknowledgments

The research is supported by the National Key R&D Program of China (2020YFB1006001), the National Natural Science Foundation of China (62176269, 62006252), the Key-Area Research and Development Program of Guangdong Province (2020B010165003), and the Innovative Research Foundation of Ship General Performance (25622112).

References

[1] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” 2017.

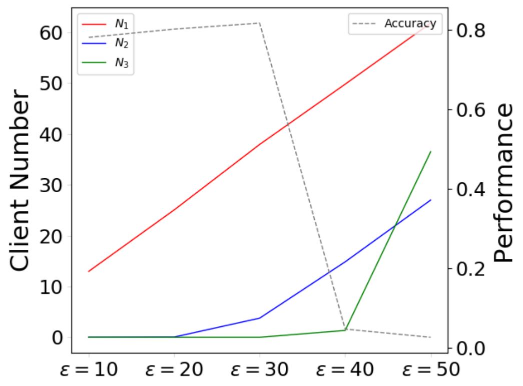


Fig. 10: Number of malicious clients in the training process, where N_1 denotes the number of malicious training clients, N_2 denotes the malicious committee clients and N_3 denotes the malicious aggregation clients.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, ser. Proceedings of Machine Learning Research, A. Singh and X. J. Zhu, Eds., vol. 54. PMLR, 2017, pp. 1273–1282.

[3] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *CoRR*, vol. abs/2003.02133, 2020.

[4] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries:byzantine tolerant gradient descent," 2017.

[5] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 1605–1622.

[6] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 2512–2520.

[7] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2021.

[8] K. A. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," *CoRR*, vol. abs/1611.04482, 2016.

[9] C. Yang, J. So, C. He, S. Li, Q. Yu, and S. Avestimehr, "Lightsecagg: Rethinking secure aggregation in federated learning," *CoRR*, vol. abs/2109.14236, 2021.

[10] Y. Hu, W. Xia, J. Xiao, and C. Wu, "GFL: A decentralized federated learning framework based on blockchain," *CoRR*, vol. abs/2010.10996, 2020.

[11] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-robust federated machine learning through adaptive model averaging," 2019.

[12] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 5650–5659.

[13] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, Dec. 2017.

[14] F. Sattler, K.-R. Müller, T. Wiegand, and W. Samek, "On the byzantine robustness of clustered federated learning," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 8861–8865.

[15] K. Yadav and B. B. Gupta, "Clustering algorithm to detect adversaries in federated learning," 2021.

[16] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," 2020.

[17] P. K. e.t., "Advances and open problems in federated learning," 2021.

[18] C. Pappas, D. Chatzopoulos, S. Lalis, and M. Vavalis, "Ipls : A framework for decentralized federated learning," 2021.

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

[20] A. Lalitha, O. C. Kilinc, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," 2019.

[21] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," 2019.

[22] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *Distributed Applications and Interoperable Systems*, J. Pereira and L. Ricci, Eds. Cham: Springer International Publishing, 2019, pp. 74–90.

[23] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279–1283, 2020.

[24] Y. Qu, L. Gao, T. H. Luan, Y. Xiang, S. Yu, B. Li, and G. Zheng, "Decentralized privacy using blockchain-enabled federated learning in fog computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5171–5183, 2020.

[25] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.

[26] X. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu, "Flchain: A blockchain for auditable federated learning with trust and incentive," in *2019 5th International Conference on Big Data Computing and Communications (BIGCOM)*, 2019, pp. 151–159.

[27] J. Konečný, B. McMahan, and D. Ramage, "Federated optimization:distributed optimization beyond the datacenter," 2015.

[28] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," 2020.

[29] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2020.

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

[31] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–9.

[32] F. Haddadpour and M. Mahdavi, "On the convergence of local descent methods in federated learning," *CoRR*, vol. abs/1910.14425, 2019.

[33] C. T. Dinh, N. H. Tran, M. N. H. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, 2021.

[34] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Trans. Parallel Distributed Syst.*, vol. 31, no. 8, pp. 1754–1766, 2020.

[35] S. U. Stich, "Local sgd converges fast and communicates little," 2019.

[36] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD," in *Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019*, A. Talwalkar, V. Smith, and M. Zaharia, Eds. mlsys.org, 2019.

[37] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, ser. Proceedings of Machine Learning Research, A. Singh and X. J. Zhu, Eds., vol. 54. PMLR, 2017, pp. 1273–1282.

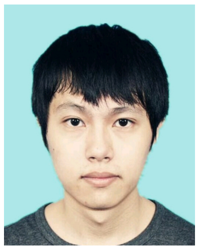
[38] Castro, Miguel, and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems*, vol. 20(4), pp. 398–461, 2002.

[39] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric)," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, 2017, pp. 253–255.

[40] H. T. Nguyen, V. Sehwag, S. Hosseinalipour, C. G. Brinton, M. Chiang, and H. Vincent Poor, "Fast-convergent federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 201–218, 2021.
[41] C. T. Dinh, N. H. Tran, M. N. H. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 398–409, 2021.
[42] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1754–1766, 2020.
[43] M. M. Amiri, D. Gunduz, S. R. Kulkarni, and H. V. Poor, "Convergence of federated learning over a noisy downlink," 2020.
[44] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.
[45] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, "Stochastic variance reduction for nonconvex optimization," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 314–323.
[46] R. Ward, X. Wu, and L. Bottou, "AdaGrad stepsizes: Sharp convergence over nonconvex landscapes," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 6677–6686.
[47] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.
[48] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.
[49] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *CoRR*, vol. abs/2010.01243, 2020.
[50] M. M. Amiri, D. Gündüz, S. R. Kulkarni, and H. V. Poor, "Convergence of update aware device scheduling for federated learning at the wireless edge," *IEEE Transactions on Wireless Communications*, vol. 20, no. 6, pp. 3643–3658, 2021.
[51] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," 2019.
[52] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *Processing*, vol. 150, 01 2009.
[53] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.
[54] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, ser. Proceedings of Machine Learning Research, A. Singh and X. J. Zhu, Eds., vol. 54. PMLR, 2017, pp. 1273–1282.
[55] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 1544–1551.
[56] L. Muñoz González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 27–38.
[57] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," *CoRR*, vol. abs/1905.06731, 2019.
[58] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," *CoRR*, vol. abs/1908.07782, 2019.

Chunjiang Che is currently working toward the B.E. degree in the School of Computer Science, Sun Yat-sen University, Guangzhou, China. His recent research interests include blockchain, federated learning and optimization.

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.



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.

Xiaoyu He received the B.Eng. degree in computer science and technology from Beijing Electronic Science and Technology Institute, Beijing, China, in 2010, the M.Sc. degree in public administration from South China University of Technology, Guangzhou, China, in 2016, and the Ph.D. degree in computer science from Sun Yat-sen University, Guangzhou, in 2019. He is currently a postdoctoral fellow at School of Data and Computer Science, Sun Yat-sen University. His research interests include evolutionary computation 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.