Computer Science and Information Systems 22(4):1457–1482 https://doi.org/10.2298/CSIS241119056W

Federated Learning with Committee Mechanism for Class Imbalance

Lang Wu and Yi Dong

School of Applied Science Beijing Information Science and Technology University, 102206
Changping District, Beijing, China
wulang@bistu.edu.cn
19862140675@163.com

Abstract. Federated learning is a collaborative machine learning approach where multiple clients train a global model without sharing raw data. Federated learning has high application value in the fields of IoT, healthcare, and others due to its decentralized data processing and privacy protection features. Despite its advantages, the classic federated learning algorithm, Federated Averaging (FedAvg), faces some limitations that affect its optimization speed and compromise system security. This paper introduces FedCCSM, a federated learning framework designed to address class imbalance and malicious client behavior. Firstly, to accelerate model optimization, a client selection mechanism is introduced based on specific criteria, ensuring a high-quality data or powerful computational clients participate in the aggregation process. This speeds up optimization and improving overall efficiency. Secondly, the adoption of a committee mechanism involves selecting a client committee to screen the model before aggregation, enhancing system security. This committee serves as a precautionary measure to prevent malicious clients from conducting adversarial attacks by intentionally providing inaccurate updates or compromising the integrity of the global model integrity. By doing so, the security and reliability of the global model are ensured throughout the collaborative learning process. Thirdly, by simulating mechanisms for unbalanced clients, the algorithm's practical application effectiveness is strengthen. Experiments on MNIST and CIFAR-10 datasets demonstrate that FedCCSM improves accuracy on imbalanced datasets by 3% compared to FedAvg and reduces the influence of malicious clients by 5%. These results highlight the potential of FedCCSM in enhancing federated learning robustness and fairness in security-sensitive applications.

Keywords: Federal Learning, Committee, Imbalance DateSet, Transcendence Coefficient, Reliability Value Criterion.

1. Introduction

As machine learning and artificial intelligence continue to shape industries worldwide, the evolution towards intelligence is evident. Traditional centralized machine learning algorithms[1][27] necessitate users to upload their local data to a central server in exchange for high-quality machine models. However, relinquishing control over data raises concerns regarding security and privacy, potentially violating user personal interests and information security. The General Data Protection Regulation (GDPR), implemented by the European Union in 2018, underscore the importance of standardized information technology practices and the establishment of a secure cyberspace environment to cornerstone

big data development and implementation. In the machine learning domain, a distributed learning framework, not requiring users to disclose private data, is targeted; yet, it can still achieve model training, known as Federated Learning (FL)[28].

FL is a decentralized machine learning approach. In Fig1, a single round of FL mainly follows the following four steps:

- First, the central server initializes the global model and sends it to the participants;
- Second, the participants train their local models using their local data, producing local model parameter updates;
- Third, the participants send the parameter updates to the central server;
- Fourth, the central server aggregates the parameter updates from all participants to generate the new parameters of the global model, and sends them back to participants.

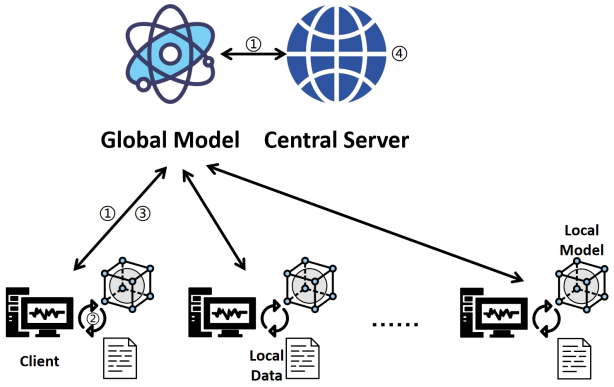


Fig. 1. Federal Learning Workflow Diagram

This process iterates continuously, allowing the global model to be optimized and enhanced without exposing individual data.

Federated learning, as a decentralized machine learning approach, offers the advantage of enabling model training across devices and organizations without centralizing data, thus safeguarding data privacy. This decentralization approach also mitigates data transfer and communication costs. Additionally, federated learning integrates data from diverse sources, enhancing the model's generalization capabilities, and facilitates local model updates, thereby improving training efficiency. Compared to centralized model training, federated learning better accommodates dispersed data sources and privacy protection requirements in real-world scenarios, presenting wide-range application.

As of now, there have been significant advancements and innovations in federated learning. Federated Averaging Algorithm (FedAvg), proposed by McMahan, is a practical method for joint learning of deep networks through iterative model averaging[28]. Zhou introduced FedGAM, an innovative federated learning algorithm designed to address client drift issues. It does so by introducing gradient norm perception minimization to achieve a locally flat loss function shape and utilizes control variables to correct local

updates, effectively solving the global flatness problem[37]. FedProx, as proposed by Li et al., addresses heterogeneity issues in federated networks by serving as a generalization and re-parameterization of FedAvg[24]. Moreover, WFB is a watermarking-based copyright protection framework for federated learning models that leverages blockchain technology to ensure model ownership and prevent unauthorized usage [33]. In addition, FedBN, introduced by Li et al, employs local batch normalization to alleviate feature shift before model averaging, thereby accelerating convergence compared to FedAvg[25]. FedFast, proposed by Muhammad et al, aims to accelerate distributed learning, achieving good accuracy for all users early in the training process and benefiting from reduced communication costs and improved model accurate[29]. Zhou et al proposed a new hierarchical FL framework RoPPFL, a robust aggregation edge FL framework tailored for computing applications. It supports privacy-preserving hierarchical FL and is resistant to poisoning attacks[40]. Moreover, Wei et al introduced a zeroth-order stochastic FL method based on Nesterov's zeroth-order (gradient-free) technique, considering both constant and diminishing step size strategies[36]. In addition, Pedrycz et al advocated for expressing Machine Learning (ML) construction results' credibility in terms of information granularity, extending the scope of FL evaluations[31]. Nergiz et al converted several classic DL methods, including the Big Transfer model, into federated versions[30]. Du et al proposed a Network Intrusion Detection algorithm (NIDS-FLGDP) based on Gaussian differential privacy federated learning[10]. Added to that, Zhang et al developed a platform architecture for a blockchain-based Industrial Internet of Things (IIoT) fault detection FL system, along with a novel Centroid Distance Weighted FedAvg (CDW_FedAvg) algorithm[9]. These advancements enhance federated learning 's efficacy and applicability.

Currently, the primary obstacle hindering the practical deployment of FL systems are their susceptibility to attacks from malicious clients[15] and impacts by imbalance data. In such systems, the central server lacks control over clients' behavior and access to their private data. Therefore, malicious clients can deceive the server by sending modified and harmful model updates, launching adversarial attacks on the global model[2]. Two types of adversarial attacks are prevalent: non-targeted attacks [6] and targeted attacks. The former aims to degrade overall model performance, causing the model to produce incorrect predictions without specifying a specific target category. This type of attack is considered as a Byzantine attack, leading to deteriorating model performance or training failure[22]. As for targeted attacks[9][6][39], they are specific, aiming to modify the model's behavior on particular data instances chosen by the attacker, such as misclassifying an image of a cat as a dog while keeping the model's performance unaffected on other data instances. Therefore, this attack requires defining a specific target category to misclassify the input as the specified target category. Both types of attacks can have catastrophic consequences, underscoring the importance of promptly detecting malicious attackers and remove their models from the FL algorithm. Defense against Byzantine attacks has been extensively researched in distributed ML. For instance, Chen proposed a variant of the classic gradient descent method based on geometric median averaging of gradients. Firstly, the parameter server groups the received gradients into non-overlapping batches to increase the similarity of non-Byzantine batches and then applied the median of batch gradients to mitigate the impact of Byzantine machines[8]. Moreover, Blanchard et al. introduced Krum, formulating the tolerance properties of aggregation rules; it is the first provably Byzantine-fault-tolerant distributed SGD algorithm[4]. In addition, Fung et

al. describes a novel defense method called FoolsGold, which is used to identify poisoning sybils based on the diversity of client updates in the distributed learning process. This system does not limit the expected number of attackers, requires no auxiliary information outside of the learning process, and makes fewer assumptions about clients and their data [11]. Furthermore, Han proposed three new robust aggregation rules for distributed synchronous Stochastic Gradient Descent (SGD) under a general Byzantine failure model, where attackers can randomly manipulate the data transferred between the servers and the workers within the Parameter Server (PS) architecture[13]. Finally, Yin developed a median-based distributed learning algorithm, achieving optimal statistical performance, better communication efficiency, and provable robustness while requiring just one communication round[38]. Obviously, the impact of malicious attacks is severe. The above is basically based on the improvement of the algorithm, yet it cannot completely block malicious attacks.

In previous research on federated learning, there are limitations and challenges in addressing class imbalance and client data integrity. Some studies focus on addressing the challenges posed by class imbalance but fall short in considering client data integrity and privacy protection. Other studies concentrate on client data integrity but often overlook the impact of class imbalance on model performance. Therefore, there are still gaps and deficiencies in research on addressing class imbalance and client data integrity in federated learning. In this context, integrating defense against malicious attack and addressing data imbalance in FL is a logical step forward. To this end, the proposal of Federated Learning with Committee Consensus and Selection Mechanism (FedCCSM) is significant[19]. The inclusion of a committee mechanism offers an effective means to detect malicious clients, thereby bolstering the security and resilience of the FL system. By implementing the committee mechanism, decisions regarding the acceptance of model updates from a participant can be made through methods like voting, effectively preventing malicious clients from impacting the system. According to the data characteristics of the client data set, the weight is weighted to enhance the anti-unbalance performance of the model. Moreover, the committee mechanism can incorporate security checks and validation mechanisms, such as data authenticity verification and model parameter legitimacy, further fortifying the system's security. Therefore, FedCCSM stands poised to effectively identify and counteract malicious client behavior while protecting participant data privacy, thereby enhancing the overall security and trustworthiness of the FL system. This approach holds significant application value across various fields such as healthcare, finance, smartphones, and IoT devices, paving the way for widespread adoption of FL in real-world scenarios.

Through studying the above questions, we will introduce new mechanisms into federated learning for improvement. As a result, the major contributions of this work can be summarized as follows:

- Improved global model training speed is achieved through the introduction of a boxplot coefficient screening mechanism. This method involves selecting high-accuracy client models for aggregation into the global model in the subsequent round of FL. By excluding relatively low-accuracy models, this approach accelerates the convergence speed of the global model;
- Incorporation of a committee consensus mechanism to detect and exclude malicious client models. Committee members assess and validate the model parameters sub-

mitted by clients to ensure they are non-malicious and capable of enhancing test accuracy. Only models meeting the criteria are allowed to participate in model aggregation, effectively screening out malicious clients;
- Implementation of a committee member election mechanism aimed at ensuring the integrity of elected committee members. Through predefined election criteria, clients failing to meet the specified standard are identified as malicious and barred from participating in client elections. This empowers the committee to effectively distinguish and exclude malicious models from participation;
- Development of a simulation for imbalanced datasets to emulate real-world client behavior. Since real client datasets frequently exhibit imbalances, which can impede training progress and undermine model efficacy, our simulation employs imbalanced datasets to assess the performance of federated learning algorithms accurately.

2. Related Work

2.1. Committee Mechanism

The committee mechanism[30] refers to a method of integrating and coordinating multiple independent ML models to improve overall predictive performance. In the committee mechanism, each model is trained independently, and the final prediction is derived from the combined voting or weighted average of all models. This integration method helps overcome the limitations of individual models, thereby enhancing prediction accuracy and robustness. The committee mechanism finds widespread application across diverse fields including financial risk assessment, medical diagnosis, natural language processing. Its advantages lie in its capability to leverage the strengths of multiple models, reduce overfitting risks, and improve generalization, demonstrating a strong adaptability in handling complex and high-dimensional data.

Therefore, the committee mechanism plays a pivotal role in enhancing the performance of ML models across various application scenarios, especially in FL[7]. Referring to Algorithm 1, in a FL training session, the committee mechanism orchestrates the collaborative process. Initially, the steps involve initializing the global model, followed by participants downloading the global model and training their local models using their local data to generate parameter updates. Subsequently, these updates are then transmitted to the committee mechanism, which aggregates them to derive new parameters for the global model. Finally, the committee mechanism distributes the aggregated global model parameters to participants for updating their local models. This iterative process facilitates FL, allowing for the optimization and enhancement of the global model while safeguarding data privacy.

2.2. Consensus Mechanism in Blockchain

The consensus mechanism[20], originated from the Byzantine Generals' Problem, describes a trust and consistency problem in a distributed system. This problem involves ten small countries surrounding a large country, with at least more than half of the small countries requiring to participate in the siege to achieve victory. However, if betrayal happens during the attack, the invaders may be annihilated. Therefore, each small country

Algorithm 1 Federated Learning Committee Mechanism

```
procedure (Global Model Initialization)
    global_model = initialize_model()
    for each round of training do
        for each participant do
            train_local_model = global_model
            local_model = train_local_model(local_data)
            local_parameters = get_parameters(local_model)
            send_parameters_to_committee(local_parameters)
        end for
        Committee Aggregation:
        global_parameters = aggregate_parameters_from_committee()
        global_model.update_parameters(global_parameters)
        Model Distribution:
        send_global_model_to_participants(global_model)
    end for
end procedure
```

does not trust the others. This example is similar to the need for nodes in a distributed system to reach a consensus regarding a decision. However, if there is a possibility of unfaithful behavior among the nodes in the system (i.e., betrayal), a consensus mechanism is required to ensure that a consensus decision can still be reached even in the presence of such unfaithful nodes.

Within the consensus mechanism, nodes can be divided into block-producing nodes, validating nodes, and accounting nodes (in Fig2). Moreover, the nodes responsible for proposing blocks are called block-producing nodes, also known as block producers, accountants, leaders, master nodes, or proposers. However, the nodes responsible for validating blocks are known as validating nodes, also called validators or backup nodes. Validating nodes must verify the legitimacy of the block producers and the blocks, as well as the correctness of the signatures. Finally, the nodes responsible for maintaining the blockchain database are called accounting nodes. Such nodes must store all blocks and verify them. Block-producing nodes, validating nodes, and accounting nodes are collectively referred to as consensus nodes. Therefore, the consensus mechanism main process includes electing block producers, proposing blocks, validating blocks, and updating the blockchain[26]. In each round, firstly a new block producer is elected. Then, the block producer proposes a block (packaging legitimate transactions from the network into a new block). Subsequently, validators verify the legitimacy of the new block. Finally, the accounting node prescribes the newly agreed block into the local database end to update the blockchain.

In the current research on federated learning, FedAvg, as a commonly used optimization method, is widely applied in the model aggregation process. However, FedAvg has certain shortcomings in terms of model security. Specifically, due to the use of a simple average aggregation method, FedAvg poses risks of privacy leakage and model tampering, which could potentially threaten the overall security of federated learning systems. To address the security deficiencies of FedAvg, the method proposed in this study combines committee mechanisms and consensus mechanisms. By introducing committee mecha-

nisms, each participant forms an independent committee during the model update process, where committee members supervise and verify each other, enhancing the reliability and security of model updates. Additionally, through the introduction of a consensus mechanism, participants must reach a consensus before submitting the model update results to the central server, ensuring the consistency and trustworthiness of model updates. This federated learning approach that combines committee and consensus mechanisms not only enhances model security but also effectively improves model performance and convergence speed, bringing new insights and opportunities for the development of federated learning systems.

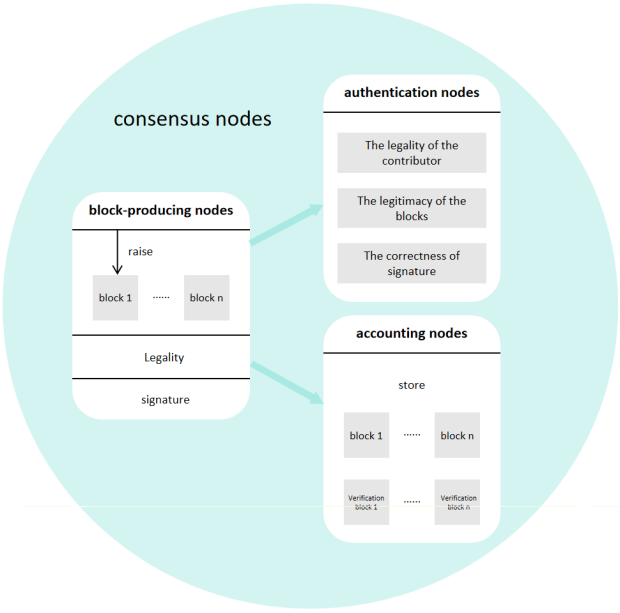


Fig. 2. Consensus Mechanism

2.3. Federal Learning and Imbalanced Dataset

Federated learning[28] is a decentralized machine learning approach that enables multiple edge devices or clients to collaboratively train a shared global model without exchanging raw data. Instead, model updates are computed locally on each device using its own data, and only the encrypted or aggregated updates are sent to a central server for aggregation. This privacy-preserving technique allows for efficient model training while protecting the privacy and security of sensitive data, making federated learning ideal for applications in healthcare, finance, and other industries where data privacy is a top priority.

Data imbalance is a common and real challenge in federated learning, like FedIBD[14]. Due to the potential increase in computational resource requirements and time costs associated with asynchronous learning, FedIBD may face challenges in performance stability

and model generalization when dealing with imbalanced data. Additionally, data privacy concerns and deployment complexities are limitations of FedIBD that require further research and improvement to enhance the system's reliability and scalability. To address the issue, we need to simulate real-world data. In the simulation test, it is a key to grasp the data imbalance. A class-imbalanced dataset[3] refers to a dataset where the number of samples in each class differs significantly, leading to a classification problem. In such datasets, the number of samples in certain classes may be much larger than in others, resulting in an uneven data distribution. Based on the total sample size, representing the classification standard, class-imbalanced datasets are divided into globally-balanced locally-imbalanced and globally-imbalanced locally-imbalanced datasets. Referring to Fig3(b), the total data quantity for each client is balanced. However, data distribution for different clients highlights a locally imbalanced state, representing the globally-balanced locally-imbalanced type. Referring to Fig3(c), the total quantity of data for each client is imbalanced, and their distribution across the different data categories is also imbalanced. Even in cases where the overall data is of a minority class, it may be the majority class locally; for instance, data k has the most data on client A, but is in the middle overall, while data j is in the majority class overall, but does not appear for client A.

The data class imbalance phenomenon is common in several real-world applications, such as rare diseases in medical diagnosis and in financial fraud detection. Moreover, the existence of class-imbalanced datasets can affect the training and performance of ML models, as they tend to predict the classes with more samples, while neglecting those with fewer samples. Therefore, the unbalance of data sets should be emphasized in federated learning.

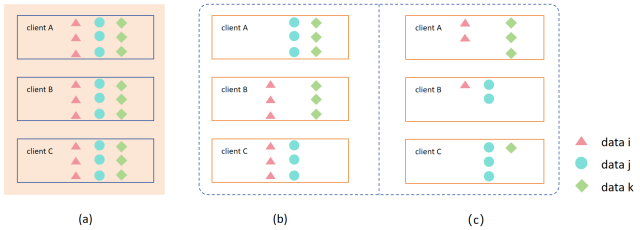


Fig. 3. Balanced and Imbalanced datasets

2.4. Malicious Client Detection

Malicious Client Detection[5][16] is a critical issue in Federated Learning, aiming to ensure that the global model is not compromised by malicious clients in a collaborative multi-party environment. Several studies have focused on proposing robust detection and defense mechanisms to address the potential impact of malicious clients on model training. Li et al. provided a review of the challenges and future directions of Federated Learning, discussing how to address the interference from malicious clients[23]. Chen and Zhou proposed a robust Federated Learning algorithm that identifies and mitigates the impact of malicious clients to improve model performance[22]. Konečný et al. proposed a

Federated Learning framework for distributed optimization, pointing out the impact of malicious clients on the training process[18]. Zhu and Han provided a detailed review of attacks and defenses in Federated Learning, proposing defense strategies based on attack types[34]. Xie et al. proposed a method to enhance the robustness of Federated Learning through malicious client detection and defense mechanisms[21]. Overall, although many existing methods provide certain robustness in theory and can handle some malicious client behaviors, their detection accuracy in practical applications still has room for improvement. These issues significantly limit the applicability of existing methods in real-world scenarios, especially in tasks with high requirements for accuracy.

3. Federated Learning Based on Committee Consensus and Selection Mechanism

The classic FedAvg learning algorithm is susceptible to contamination by malicious clients [17] due to its indiscriminate aggregation of client models. Moreover, the average aggregation method can hinder overall model training speed. FedCCSM addresses these issues by integration a committee mechanism to select and evaluate clients. In such a defensive algorithm, two criteria are employed to select committee members from the client pool, asked with filtering well-trained client models. Committee members must possess the capability to score models, enabling them to control the client models participating in the aggregation rather than aggregating all models blindly. By ensuring the integrity of committee members, the probability of malicious clients disrupting the global model training process is reduced significantly. To guarantee the honesty of committee members and facilitate secure aggregation, a new committee mechanism has been devised, encompassing a scoring system, selection strategy, and election strategy. Meanwhile, to simulate the actual client dataset distribution, random sharding is utilized to simulate an imbalanced dataset, achieving the most realistic effect.

Therefore, this section will provide a detailed introduction to the proposed framework and mechanism. In this case, we assume there are C clients forming a client group $\{C_i\}_{i=1}^{C}$, and the dataset for each client is denoted as M_{C_i}. The meanings of each variable notation are in Table 1.

3.1. Allocate Client Datasets

Referring to Fig4, step I refers to allocating the training dataset and testing the dataset for each client. For a dataset M containing m samples with a total of k classes, each class containing N samples; therefore, the dataset can be represented as $M = \{N_i\}_{i=1}^{K}$. To construct an imbalanced dataset, it is required to generate first the imbalanced parameters. Typically, for a collection of data containingk classes, the number of data classes in an imbalanced dataset is randomly selected between 1 and k. Therefore, a random array S containing C elements is generated, where each element is a random number between 1 and k. This array serves as the random shard array, where each element denotes the number of shards for the corresponding client.

To create a globally balanced and locally imbalanced train dataset, the data volume for each client has been determined during the simulation. To maximize the use of dataset M, the data volume for each client is set as m/C. Therefore, each shard size for the i^{th}

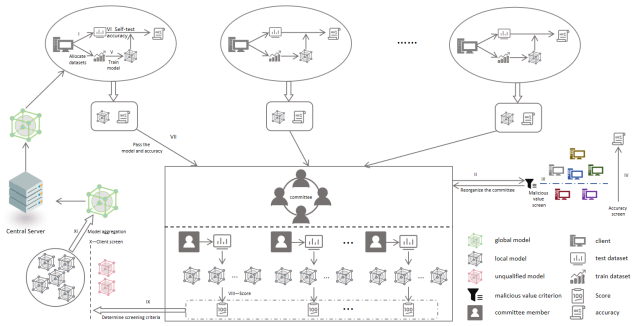


Fig. 4. FedCCSM Framework Diagram

client is m/CS_i. As a result, the entire dataset can be divided into CS_i shards. By using the np.random.permutation function to randomly permute the indices of all shards, client N_i takes the first S_i indices corresponding to the shards, ensuring data randomness. This approach leads to a more realistic simulation for data situation for several clients.

For a globally unbalanced training dataset, the construction process only requires a modification to one step of the process used for creating a globally balanced and locally imbalanced dataset. Specifically, the shard size for the entire dataset is not determined by each client but is instead uniformly defined. We denote this uniform shard size as s, which must fall between 1 and K (here, s can also serve as an indicator of the imbalance rate). Considering the scenario with the maximum data volume, the maximum data volume for each client remains m/C. Hence, the size of each shard becomes m/Cs. The subsequent steps, where each client acquires a certain number of shards, are still dictated by the array S, implying that client N_i's dataset comprises S_i different types of data, totaling mS_i/sC of data.

The algorithm also searches for malicious clients; thus, in addition to the imbalanced dataset, it is required to simulate a portion of malicious clients. To construct malicious clients, an additional step is added to the algorithm after creating the imbalanced dataset, where a portion of the information in each shard is modified to become incorrect. This study uses the Modified National Institute of Standards and Technology (MNIST) handwritten digit dataset as the basis for detection. For example, the handwritten digit data corresponding to a number x is changed to correspond to $9-x$; thus, the handwritten digit for 6 becomes 3. A client with such erroneous training data is considered a malicious client. The trained models by these clients will have significant different performances compared to those trained by normal clients, leading to the evaluation of the exact model effectiveness.

When simulating the testing dataset, it is crucial to ensure an adequate representation of each class of data. Therefore random shuffling of shards is not permissible. Instead, a portion of data from each class is sequentially selected to form the testing dataset. This step concludes the simulation of the clients.

3.2. Electing the Committee for Generation

Next, step II "reorganize the committee" is proposed. The details shown in Fig5, it mainly consists of selecting clients to form a committee G, consisting of a central server G_0 as fixed member and a number of members, where the number of members set as g. As the objective consists of detecting clients throughout the committee, the committee members should meet certain criteria. Step III involves determining the reliability of clients, while step IV focuses on sorting and selection.

This algorithm defines the reliability scoring criterion e to filter unreliable clients. Specifically, after the federated learning training is conducted by each client, each client tests and obtains a local prediction accuracy rate. During the initial selection process, the local model and the locally measured accuracy rate are transferred to the committee. The fixed member detects each client's local model to obtain the prediction accuracy rate. The absolute value of the difference between the local accuracy rate of each local model and the accuracy rate measured by the fixed member is taken as the reliability of this round. According to the magnitude of the reliability from low to high, g temporary members are selected. In subsequent rounds, the g temporary members selected in the previous round are responsible for testing the local models of all other clients in this round. Thus, each local model acquires g prediction accuracy rates. The absolute values of the differences between each of these accuracy rates and the local accuracy rate of the model are taken and then averaged to obtain the reliability f_i. According to the magnitude of the reliability from low to high, g temporary members are selected for the next round. The g temporary members selected in each round, along with the fixed member, form a $g+1$ person federated learning committee, which determines the selection criteria for the local models of each client participating in the global model aggregation.

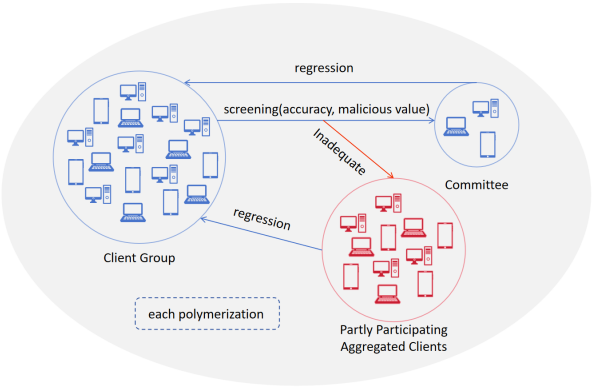


Fig. 5. The Interaction Between Clients and Committee

The remaining clients that do not meet the criteria participate in the subsequent local training steps.

Algorithm 2 Electing the Committee for Generation

Server executes:
Initialize G^0
for each round i = 0, 1, 2,... **do**
  if i mod v = 0 **then**
    for each round j = 0, 1, 2,... **do**
      $C_j \rightarrow f_j$
      if $f_j < e$ **then**
        $C_j \rightarrow G^i$
      end if
    end for
  end if
  $G_i \rightarrow C$
end for

3.3. Model Training

The focus is on non-convex neural network objectives. For machine learning problems, the typical formulation is $f_i(w) = l(x_i, y_i; \omega)$, representing the loss incurred by using model parameter ω to predict on example (x_i, y_i). It is assumed that there are K clients participating in this round of training, where P_i is the set of indices of data points on client C_i, and $n_i = |P_i|$. Therefore, the algorithm considered applies to the following form of finite-sum objective[22]:

$$\min_{\omega \in R^d} \mathrm{f}(\omega) \quad where \quad \mathrm{f}(\omega) = \sum_{i=1}^{K} \frac{n_i}{n} \mathrm{F_i}(\omega) \quad where \quad \mathrm{F_i}(\omega) = \frac{1}{n_i} \sum_{j \in P_j} \mathrm{f_j}(\omega). \tag{1}$$

Consequently, we proceed to step V, involving model training. For each client C_i, the parameters of the global model $\omega^{(l)}$(the l^{th} round) are first loaded into the client's model. Then, the client's data loader is created based on the client's own training dataset where each client undergoes d rounds of local training. In each round, a pair of data and labels is retrieved from the training set and iterated upon. Firstly, data and labels are moved to the device for computation; in such case, the CPU is deployed. The data is then fed into the neural network.

Secondly, Refer to Fig6, a Convolutional Neural Network (CNN) is established with two 5*5 convolutional layers[32], with 32 and 64 channels, respectively. Each layer is followed by a 2*2 max-pooling layer. This is connected to a fully connected layer with 512 units and ReLU activation, and a final softmax output layer (totaling 1,663,370 parameters).

Next, the reshaping of the input data into a tensor of shape (-1, 1, 28, 28) is performed. The process expression for the first convolutional layer is as follows:

$$h^{(l)} = \sigma(\omega^{(l)} * h^{(l-1)} + b^{(l)}). \tag{2}$$

Then, applying the ReLU the activation function $f(x) = max(0, x)$ is performed. Following this process, a pooling layer is applied, followed by another convolutional layer, an activation function, and another pooling layer, resulting in the tensor being flattened

into a one-dimensional shape having a size of (-1, 7*7*64). Consequently, the flattened tensor is passed through the first fully connected layer, followed by another ReLU activation function. Then, the result is passed through the second fully connected layer to generate the final output. As a result, the output consists the predicted result.

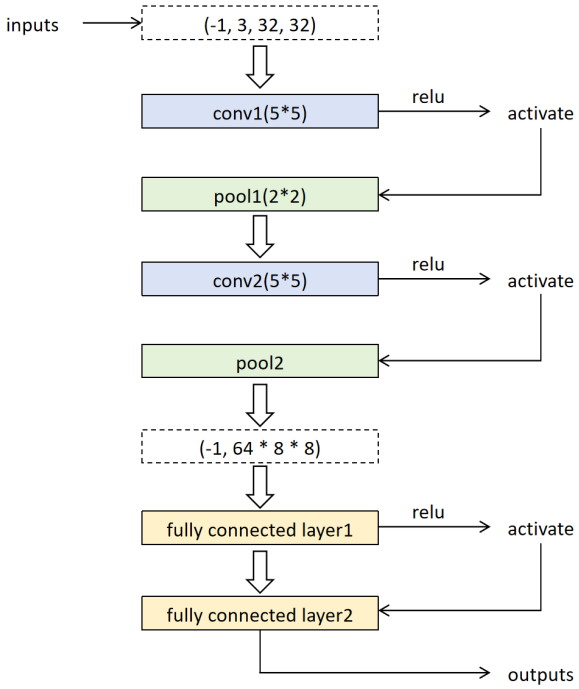


Fig. 6. The steps of Convolutional Neural Network

Thirdly, the loss function is calculated between the predicted results and the true values using the cross-entropy loss function. As the MNIST dataset essentially represents a ten-classification problem, with labels being arrays of length ten where the i^{th} element is equal to the unit and the rest are null, we use the cross-entropy loss function to calculate the loss. If i represents the sample, y denotes the actual label, a indicates the predicted output, and n highlights the total number of samples; then, the loss value is written as follows:

$$loss = -\frac{1}{n}\sum_{i} y_i \ln p_i \,. \tag{3}$$

Fourthly, the backpropagation is performed to calculate the gradients. Fifthly, the Stochastic Gradient Descent (SGD) algorithm is utilized as the optimizer to update the model's parameters. For each i, the algorithm expression is defined as follows:

$$\theta_j := \theta_j + \alpha(y^{(i)} - \mathrm{h}_\theta(x^{(i)}))x_j^{(i)} \quad where \quad \mathrm{h}_\theta(x) = \theta_0 + \sum_{i=1}^{n} \theta_i x_i \,. \tag{4}$$

Finally, after the local training is completed, the client model is updated with the latest model, and the accuracy of the model is measured with the local test data set in Step VI.

3.4. Model Selection and Aggregation

After each client C_i completes his local training in this round, he obtains the latest model ω_i^l and evaluates its accuracy on the local test set, denoted as a_i.

Step VII involves transmitting the trained local updated model parameters and accuracy to the committee. Each committee member G_i and central server G_0 receives all updated model parameters and their accuracies to prepare for the next step.

Step VIII contains the committee members scoring for each client. The members of the committee test the local models of the other clients based on their local test datasets. Each local model of the client will obtain g test values.

In step IX, by differentiating these test values with the accuracy of the client's own measurement and averaging the absolute value of the difference, the reliability e_j of the client's local model can be determined. Each model $\omega_j^{(l)}$ corresponds to a score set include reliability and accuracy. To prevent unreliable clients from infiltrating the committee and maliciously changing the scores given to the models, all values are analyzed in a unified centralized analysis to determine the reliability criteria e. (described in section 4.1).

Then, the accuracy rates of the selected local models of the clients are sorted, and the upper and lower quartiles and the interquartile range are calculated to determine the accuracy standard line k, as follows:

$$k = q_1 - t * iqr \,. \tag{5}$$

where t is a tunable multiple of iqr, known as exceedance coefficient.

Step X is represented in Figure 3. Exclude clients with accuracy below k and reliability above e[12]. The models' parameters that meet both criteria are aggregated, corresponding to step XI.

$$\theta_i^{(l+1)} = \frac{1}{n+1} \sum_{i=0}^{n} \theta_i^{(l)} \,. \tag{6}$$

Finally, in step XII, the aggregated global model $w^{(l+1)}$ is transmitted to the Central Server. In this way, a complete round of federated learning process is achieved.

4. Experimental Results

The experimental testing is divided into two phases based on the imbalanced dataset, namely, all normal simulation and the introduction of malicious clients. Before starting the experiment, it is important to determine the key coefficients.

Table 1. Notation Explanation

Notation	Description
C_i	Client with the index i
M_{C_i}	Dataset of client i
N_i	Data of the i^{th} category
Dr	Train dataSet
De	Test dataSet
S	Randomly partitioned array
S_i	The i^{th} shard number
s	Uniform shard number(also an indicator for the imbalance rate)
K	The number of clients participating in each round of training
G_i	The i^{th} committee member
G_0	fixed central server
H_i	The i^{th} candidate member
P_k	The set of indices of data on client k
t	Exceedance coefficient
ω	General model parameters
$\omega^{(l)}$	The l^{th} round global model
$W^{(l)}$	filter
$\omega_i^{(l)}$	The local updated model of the i^{th} client in the l^{th} round
a_i	The accuracy of client i's self-assessment
d	epoches
e	malicious standard
a_i^j	The accuracy of client i in detecting $\omega_i^{(l)}$
f_j	The maliciousness of client j
θ_j	The parameters
k	Accuracy standard
t	Threshold multiplier
iqr	Interquartile range
$q_{1/3}$	Lower/upper quartile
p_i	predicted output
m	maliciousness criterion
u	committee number
v	model validation frequency of communications

Algorithm 3 Federated Learning Based on Committee Consensus and Selection Mechanism

Server executes:
Initialize ω^0
for each round i = 1, 2,... **do**
    W $\leftarrow C_i(e_i > e)$
    $C_F \leftarrow$ (random set of m clients)
    for each client j $\in C_F$ **do**
        $w_j^{(l)}$ = ClientLocalUpdate(j, $\omega_j^{(l-1)}$)
        $a_j = TestClientAccuracy(C_De, \omega_j^{(l)})$
    end for
    for each Committee member in W_i **do**
        $a_i^j = TestClientAccuracy(\omega_j^{(l)})$
    end for
    $f_j = \sum_{k=1}^{w} a_i^j$
    q1, q3, iqr $\leftarrow$ boxplots(a_i)
    k = q1 - t * iqr
    $\omega^{(l+1)} \leftarrow Aggregation(k, e, \omega_{C_F}^l)$
end for

ClientLocalUpdate: // Run on client k
$\beta \leftarrow$ (split M_k into batches of size B)
for each local epoch i from 1 to e **do**
    for each batch b $\in \beta$ **do**
        $\omega_j^{(l+1)} \leftarrow \omega_j^{(l)}$ - $\eta(\omega_j^{(l)}$; b)
    end for
end for
return $w_j^{(l+1)}$ to server.

4.1. Coefficient Adjustment and Determination

This algorithm involves two parameters: the transcendence coefficient t and the reliability value criterion, both playing a crucial role in the client screening step. The transcendence coefficient t determines the standard line for screening accuracy, i.e., the lower accuracy bound. Experiments are conducted with t=0.3,0.4,0.5,0.6 and 0.7 as the imbalance rates to generate the accuracy of the global model while keeping other parameters constant.

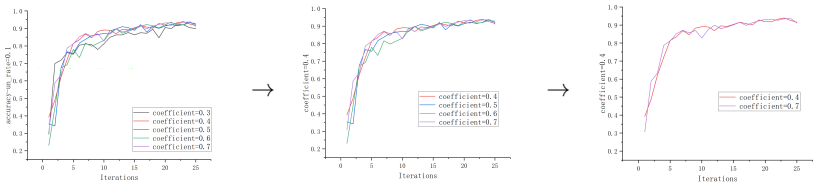
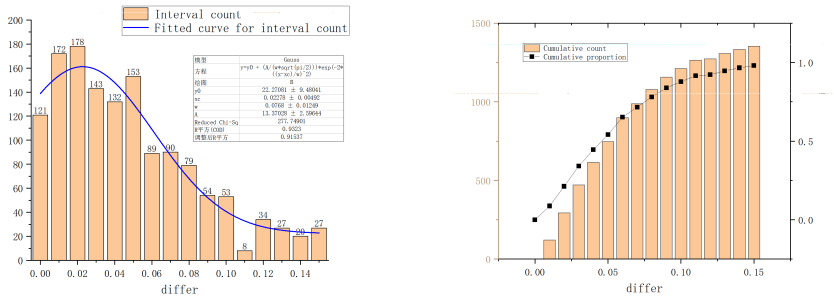


Fig. 7. Experimental Results with Different Transcendence Coefficient

Referring to Fig7, the algorithm performs best when t=0.4 referring to the speed of accuracy improvement and the stability during the improvement process. The reliability value criterion also applies experimental data as a reference. A large dataset is constructed by scoring clients with no malice where the normal range and values of scores are analyzed. This analysis helps determining the standard for reliability values. During the experiment, approximately 600 scores are collected from 30 rounds for two randomly selecting clients. Based on Fig8, 91.59% of clients have reliability scores less than 0.11. Therefore, the standard for screening unreliability clients in the model is set accordingly.



(a) Distribution of Differences **(b)** cumulative difference statistics

Fig. 8. Client Malice Score Distribution Chart

4.2. Global Imbalance

Our goal is to develop outstanding models to enhance the performance of devices in image classification and language modeling tasks, thereby improving user experience and device usability.

In the experiment of global balance and local imbalance, MNIST and CIFAR-10 datasets are deployed as the experimental datasets to test the effectiveness of the algorithm.

In the experiments concerning local imbalance in the dataset, a series of experiments were undertaken to investigate the model's ability to manage imbalanced classes. In this section, the dataset was partitioned into multiple subsets according to class combinations. Models were trained and assessed for each subset to analyze their performance in the context of local imbalance. Specifically, the study juxtaposed the accuracy, precision, and other metrics of the models across various subsets, along with their efficacy in identifying imbalanced classes.

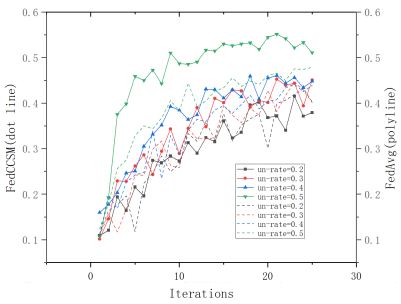
We conducted a significance t-test[35] on the accuracy of multiple experiments under the same parameters to ensure the stability of the results.

Ensuring all parameters are consistent, we obtained ten accuracy results from ten experiments: 0.9779, 0.9794, 0.9805, 0.9796, 0.9803, 0.9593, 0.9736, 0.9728, 0.9735, 0.9682. The null hypothesis (H_0) is "These values fluctuate around a mean with a small amplitude, and the average remains stable around 0.9751." The alternative hypothesis (H_1) is "These values fluctuate around a mean with a large amplitude, and the average is not stable around 0.9751." Firstly, the calculated average of these values is 0.9751. Secondly, the sample standard deviation is calculated to be 0.0065. Thirdly, the t-value is computed as

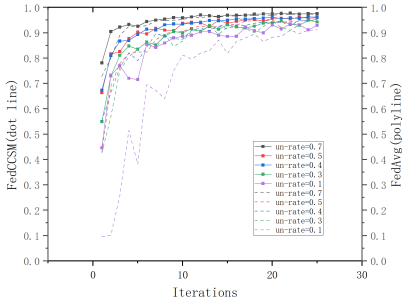
$$t = \frac{0.9751 - 0.9751}{0.0065/\sqrt{10}} = 0. \tag{7}$$

In the fourth step, the critical value for a t-distribution with 9 degrees of freedom is determined by referring to the t-distribution table. For a two-tailed test, the critical value is ±2.262. Finally, based on the calculated t-value of 0, which is less than the critical value of 2.262, we fail to reject the null hypothesis. Therefore, we accept the null hypothesis, indicating that the stability of these values around the mean of 0.9751 is good, with a small fluctuation amplitude.

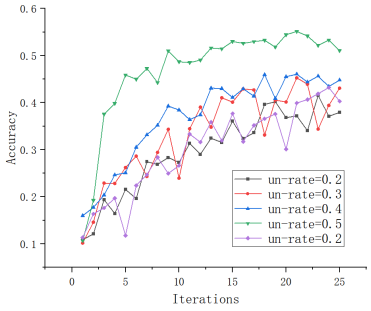
In the experimental analysis, a noticeable trend emerged wherein the improved FedCCSM model showed a swifter improvement in global model accuracy in comparison to FedAvg. Fig9(a) and (b) illustrates the phenomenon, showcasing an increase in training rounds and the accuracy of the improved model speed, while FedAvg displayed a comparatively slower progress. It is evident that on the Cifar-10 dataset, especially when the un-rate is 0.5, FedCCSM shows an overall accuracy improvement of around 5% compared to FedAvg throughout the entire federated learning process. Additionally, there are improvements of 1-2% on other un-rates as well. Furthermore, the improved model surpassed FedAvg in overall accuracy, reflecting superior precision. Additionally, the smoother curve of the improved model, characterized by minimal fluctuations, indicates a more stable response to changes in the dataset. These findings underscore the superior performance and enhanced stability of the improved model in managing imbalanced datasets, providing



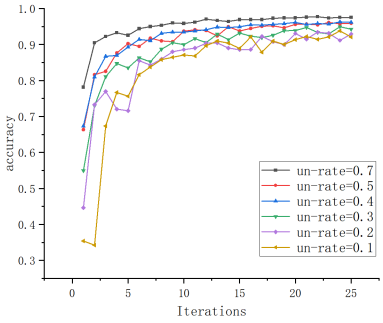
(a) FedCCSM and FedAvg in Cifar-10



(b) FedCCSM and FedAvg in Mnist



(c) FedCCSM along in Cifar-10



(d) FedCCSM along in Mnist

Fig. 9. Differ Global Imbalance Accuracy Results between FedAvg and FedCCSM in Mnist and Cifar-10

more reliable results for practical applications. Currently, the experimental results support the superior performance of the improved model over FedAvg.

Moreover, Fig9(c) and (d) corroborate the aforementioned observations, depicting a rapid improvement in accuracy with an increase in training rounds for the improved model. Despite the varying resistance encountered during model training due to different imbalance rates, as the imbalance rate increases, a trend of increased rounds is applied to reach the peak; yet, good results can be achieved.

Table 2. Comparison of methods in different hyperparameters on MNIST

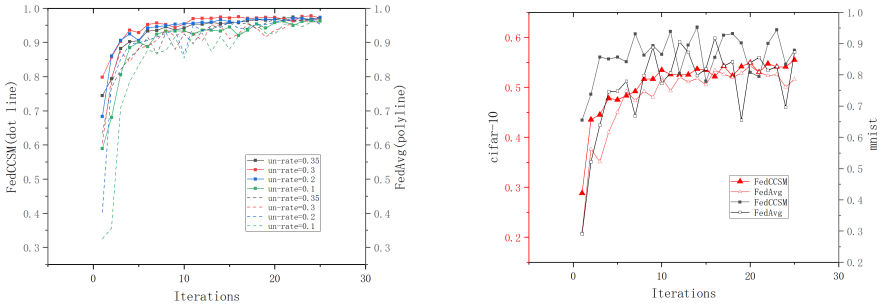
Hyperparameters value		**FedAvg**	**FedCCSM**
batchsize	$b = 8$	0.9463	0.9363
	$b = 10$ (used)	**0.9593**	**0.9499**
	$b = 15$	0.9298	0.9041
	$b = 20$	0.9090	0.8975
learning_rate	$lr = 0.001$	0.8297	0.7887
	$lr = 0.005$	0.9218	0.9052
	$lr = 0.01$ (used)	**0.9593**	**0.9499**
	$lr = 0.05$	0.9493	0.9408

Based on the experimental results and sensitivity analysis of hyperparameters, we found that in this study, the FedCCSM method demonstrates better robustness and stability. Particularly, under the conditions of batch size = 0.01 and learning rate = 10, its performance significantly outperforms the FedAvg method. Therefore, in this experiment, we used these parameters for overall experimentation and comparison. This indicates that the FedCCSM method exhibits better adaptability and stability in federated learning tasks, serving as an effective optimization method to enhance model performance and convergence speed. Hence, it is recommended to prioritize the use of the FedCCSM method in practical applications to achieve better results.

4.3. Global Balance

In the global balance experiment section, this study conducted unified training and evaluation on the entire dataset, incorporating malicious clients into the simulated client group. From the experimental data, it is evident that our proposed federated learning model, FedCCSM, demonstrates accelerated initial performance enhancement compared to the traditional FedAvg. Furthermore, in terms of the change in global model accuracy, the balance between local imbalance and global imbalance is closer to Independent and Identically Distributed (IID), as evidenced by the relatively higher accuracy.

According to Fig10(a), FedCCSM exhibits higher accuracy and a steeper slope, indicating faster convergence and superior performance during the early stages of model training. Additionally, as the training progresses, the performance improvement of FedCCSM gradually stabilizes and surpasses FedAvg, suggesting that FedCCSM can maintain stable performance in the later stages of training. Compared to FedAvg, FedCCSM demonstrates clear advantages in terms of convergence speed and performance stability.



(a) FedCCSM and FedAvg in Honest Global Balance

(b) FedCCSM and FedAvg in Malicious Device Detection

Fig. 10. Results about Global Balance in Honest and Malicious

The following experiment involved manipulating the training set based on imbalance, simulating malicious clients by introducing incorrect training sets. This was conducted to evaluate the algorithm's resistance to malicious behavior. When applied to MNIST and CIFAR-10, we intentionally disrupted the correspondence between the data and labels, resulting in mislabeled data. This adversely affected the overall training process.

The results in Fig10(b) demonstrate that the improved code outperforms FedAvg in overall accuracy, indicating greater resistance to malicious behavior. Furthermore, the curve of the enhanced code exhibits smoother trajectories with smaller fluctuations, suggesting that it can maintain accuracy more consistently when faced with malicious data. This provides more reliable results for practical applications.

Table 3. Comparison of methods on the MNIST dataset under global balance and malicious

Hyperparameters	**value**	**Methods**			
		FedAvg	**FedProx**	**MOON**	**FedCCSM(this paper)**
Reliable	un-rate=0.1	0.9599	0.9393	0.9526	0.9652
	un-rate=0.2	0.9533	0.9592	0.9621	0.9735
	un-rate=0.3	0.9601	0.9526	0.9592	0.9717
	un-rate=0.35	0.9719	0.9622	0.9637	0.9709
Malicious	Malic-ratio=0.2	0.8728	0.8841	0.8775	0.9160

The experimental results demonstrate that the FedCCSM model shows improvement in federated learning, with higher accuracy compared to traditional FedAvg, FedProx, and MOON models, regardless of the imbalance rate. In this experiment, the highest accuracy improvement was observed at an imbalance rate of 0.2, with an increase of 2.02%. When facing malicious clients with a simulated proportion of 0.2, FedCCSM showed improvements of 4.32%, 3.19%, and 3.85% compared to FedAvg, FedProx, and MOON, respec-

tively. Clearly, when dealing with a higher proportion of malicious devices, FedCCSM can better maintain model accuracy, demonstrating stronger robustness and generalization capabilities.

The aim of this study, as demonstrated through the above experiments, is to thoroughly analyze the performance of the model on malicious clients with imbalanced class datasets. Additionally, the study seeks to explore effective training strategies for addressing imbalanced class datasets and enhance the model's ability to resist malicious attacks. The results consistently demonstrate that FedCCSM has the capability to handle imbalanced datasets and resist malicious attacks.

4.4. All Balance

In order to test the algorithm's defense effectiveness against malicious attacks and compare its advantages and disadvantages with other similar algorithms, this part is based on the MNIST dataset, comparing the performance of FedCCSM with three methods, MUD-HoG, Foolgolds, and Fedavg, evaluating the performance indicators. Malicious attacks are categorized into non-targeted poisoning attacks[39] and targeted poisoning attacks, with targeted poisoning attacks including single-label flip attacks[8], multi-label flip attacks, and backdoor attacks[4][11]. We selected backdoor attacks as the type of attack to test the algorithm's performance. Backdoor attacks involve attackers implanting trigger patterns (backdoor triggers) in certain training/testing data to inject a backdoor, causing the model to make incorrect judgments on data with specific features. The experiment initializes the number of clients to 40, including 50% malicious clients. A federated learning model is trained over 200 communication rounds with 10 local training rounds.

Table 4. Comparison of methods on the MNIST dataset under iid distribution

Methods	Backdoor Accuracy	Model Testing Accuracy
Fedavg	99.3	97.9
Foolsgold	99.6	98.3
MUD-HoG	82.5	98.4
FedCCSM(this paper)	11.6	98.7

The experiment evaluated the effectiveness of defense methods from two aspects: model testing accuracy and backdoor accuracy. Model testing accuracy refers to the accuracy of the global model on the test set. Backdoor accuracy assesses adversarial backdoor training, measuring the number of injected samples classified as the attacker's target label. If a client's sample with a backdoor attack is predicted as the malicious target, it is considered successful in identifying and defending against the attack on that client. Depending on the exclusion of clients with backdoor attacks, a decrease in backdoor accuracy implies a reduction in the number of clients carrying backdoor attacks in the client group. Therefore, as the algorithm trains the model towards the later stages, a low value of backdoor accuracy indicates the exclusion of more backdoor attacks, suggesting that the algorithm provides the best defense against backdoor attacks.

Table 2 shows the performance results on the MNIST dataset. Compared to other defense methods, FedCCSM provides the best protection, being able to effectively eliminate

backdoor attacks to the greatest extent, demonstrating the strongest defense capability. Additionally, it improves model testing accuracy while minimizing the impact on model training.

In this study, we delve into the theoretical implications of federated learning and deep learning in the context of imbalanced datasets and privacy security. Our research findings reveal the performance differences of different algorithms in handling imbalanced datasets, providing theoretical guidance for adjusting algorithms to improve model accuracy and robustness. Additionally, our study explores the effectiveness and limitations of privacy protection technologies in federated learning and deep learning, laying a theoretical foundation for designing more secure federated learning frameworks. These discoveries offer important theoretical support for addressing challenges related to imbalanced datasets and privacy security, and provide valuable insights for future research and practical applications.

However, it is worth noting that this study also has some limitations that need further exploration and resolution. Firstly, our experimental datasets are limited to MNIST and Cifar-10, without studying more datasets. Additionally, our research is primarily focused on theoretical analysis and simulated experiments, and the practical application effects in real-world scenarios still need further validation. Therefore, ongoing attention and updates are necessary.

5. Conclusion

This paper introduces FedCCSM, a federated learning framework that addresses class imbalance and malicious client behavior through a committee-based consensus mechanism. Experiments on MNIST and CIFAR-10 datasets demonstrate significant improvements in accuracy and robustness compared to baseline methods, highlighting the effectiveness of FedCCSM. These findings contribute to advancing federated learning by improving fairness and reliability in distributed systems.

Future research should focus on exploring adaptive committee selection strategies, applying FedCCSM to diverse datasets, and assessing its scalability in real-world applications. Research can be conducted on how to make federated learning systems resilient to a wider range of malicious attacks, including data poisoning, model manipulation, and privacy breaches. This may involve the development of new security and privacy protection techniques to ensure the security and robustness of federated learning systems. Additionally, new evaluation metrics can be developed to assess performance on imbalanced datasets.

Acknowledgments. This work is supported by the Beijing Information Science and Technology University young backbone teacher program project (YBT 202447), the Future Blockchain and Privacy Computing Advanced Center Project Funding (NO.202203).

References

1. Athilakshmi, R., Jacob, S.G., Rajavel, R.: Automatic detection of biomarker genes through deep learning techniques: A research perspective. Stud. Informatics Control, Vol.32(No.2), 51–61. (2023)

2. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. In: International conference on artificial intelligence and statistics. pp. 2938–2948. PMLR (2020)
3. Batista, G.E., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD explorations newsletter 6(1), 20–29 (2004)
4. Blanchard, P., El Mhamdi, E.M., Guerraoui, R., Stainer, J.: Machine learning with adversaries: Byzantine tolerant gradient descent. Advances in neural information processing systems 30 (2017)
5. Calik Bayazit, E., Koray Sahingoz, O., Dogan, B.: Deep learning based malware detection for android systems: A comparative analysis. Tehnički vjesnik 30(3), 787–796 (2023)
6. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 ieee symposium on security and privacy (sp). pp. 39–57. Ieee (2017)
7. Che, C., Li, X., Chen, C., He, X., Zheng, Z.: A decentralized federated learning framework via committee mechanism with convergence guarantee. IEEE Transactions on Parallel and Distributed Systems 33(12), 4783–4800 (2022)
8. Chen, Y., Su, L., Xu, J.: Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. Proceedings of the ACM on Measurement and Analysis of Computing Systems 1(2), 1–25 (2017)
9. Dai, J., Chen, C., Li, Y.: A backdoor attack against lstm-based text classification systems. IEEE Access 7, 138872–138878 (2019)
10. Du, J., Yang, K.: Nids-flgdp: Network intrusion detection algorithm based on gaussian differential privacy federated learning. Journal of Circuits, Systems and Computers 33(03), 2450048 (2024)
11. Fung, C., Yoon, C.J., Beschastnikh, I.: Mitigating sybils in federated learning poisoning. arXiv preprint arXiv:1808.04866 (2018)
12. Gupta, A., Luo, T., Ngo, M.V., Das, S.K.: Long-short history of gradients is all you need: Detecting malicious and unreliable clients in federated learning. In: European Symposium on Research in Computer Security. pp. 445–465. Springer (2022)
13. Han, Y., Zhang, X.: Robust federated training via collaborative machine teaching using trusted instances. arXiv preprint arXiv:1905.02941 (2019)
14. Hou, Y., Li, H., Guo, Z., Wu, W., Liu, R., You, L.: Fedibd: a federated learning framework in asynchronous mode for imbalanced data. Applied Intelligence 55(2), 1–17 (2025)
15. Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al.: Advances and open problems in federated learning. Foundations and trends® in machine learning 14(1–2), 1–210 (2021)
16. Kavitha, S., Uma Maheswari, N., Venkatesh, R.: Intelligent intrusion detection system using enhanced arithmetic optimization algorithm with deep learning model. Tehnički vjesnik 30(4), 1217–1224 (2023)
17. Kolasa, D., Pilch, K., Mazurczyk, W.: Federated learning secure model: A framework for malicious clients detection. SoftwareX 27, 101765 (2024)
18. Konečnỳ, J., McMahan, B., Ramage, D.: Federated optimization: Distributed optimization beyond the datacenter. arXiv preprint arXiv:1511.03575 (2015)
19. Kumar, P., Kumar, R., Kumar, A., Franklin, A.A., Garg, S., Singh, S.: Blockchain and deep learning for secure communication in digital twin empowered industrial iot network. IEEE Transactions on Network Science and Engineering 10(5), 2802–2813 (2022)
20. Lashkari, B., Musilek, P.: A comprehensive review of blockchain consensus mechanisms. IEEE access 9, 43620–43652 (2021)
21. Lewis, C., Varadharajan, V., Noman, N.: Attacks against federated learning defense systems and their mitigation. Journal of Machine Learning Research 24(30), 1–50 (2023)
22. Li, S., Cheng, Y., Wang, W., Liu, Y., Chen, T.: Learning to detect malicious clients for robust federated learning. arXiv preprint arXiv:2002.00211 (2020)

23. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. IEEE signal processing magazine 37(3), 50–60 (2020)
24. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. Proceedings of Machine learning and systems 2, 429–450 (2020)
25. Li, X., Jiang, M., Zhang, X., Kamp, M., Dou, Q.: Fedbn: Federated learning on non-iid features via local batch normalization. arXiv preprint arXiv:2102.07623 (2021)
26. Li, Y., Chen, C., Liu, N., Huang, H., Zheng, Z., Yan, Q.: A blockchain-based decentralized federated learning framework with committee consensus. IEEE Network 35(1), 234–241 (2020)
27. MAIER, M.I., Czibula, G., DELEAN, L.R.: Using unsupervised learning for mining behavioural patterns from data. a case study for the baccalaureate exam in romania. Studies in Informatics and Control 32(2), 73–84 (2023)
28. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)
29. Muhammad, K., Wang, Q., O'Reilly-Morgan, D., Tragos, E., Smyth, B., Hurley, N., Geraci, J., Lawlor, A.: Fedfast: Going beyond average for faster training of federated recommender systems. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 1234–1242 (2020)
30. Nergiz, M.: Federated learning-based colorectal cancer classification by convolutional neural networks and general visual representation learning. International Journal of Imaging Systems and Technology 33(3), 951–964 (2023)
31. Pedrycz, W.: Advancing federated learning with granular computing. Fuzzy Information and Engineering 15(1), 1–13 (2023)
32. Qian, Y., Bi, M., Tan, T., Yu, K.: Very deep convolutional neural networks for noise robust speech recognition. IEEE/ACM Transactions on Audio, Speech, and Language Processing 24(12), 2263–2276 (2016)
33. Shao, S., Wang, Y., Yang, C., Liu, Y., Chen, X., Qi, F.: Wfb: watermarking-based copyright protection framework for federated learning model via blockchain. Scientific Reports 14(1), 19453 (2024)
34. Sikandar, H.S., Waheed, H., Tahir, S., Malik, S.U., Rafique, W.: A detailed survey on federated learning attacks and defenses. Electronics 12(2), 260 (2023)
35. Smucker, M.D., Allan, J., Carterette, B.: A comparison of statistical significance tests for information retrieval evaluation. In: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management. pp. 623–632 (2007)
36. Wei, L.Y., Yu, Z., Zhou, D.X.: Federated learning for minimizing nonsmooth convex loss functions. Mathematical Foundations of Computing 6(4), 753–770 (2023)
37. Xu, Y., Ma, W., Dai, C., Wu, Y., Zhou, H.: Generalized federated learning via gradient norm-aware minimization and control variables. Mathematics 12(17), 2644 (2024)
38. Yin, D., Chen, Y., Kannan, R., Bartlett, P.: Byzantine-robust distributed learning: Towards optimal statistical rates. In: International conference on machine learning. pp. 5650–5659. Pmlr (2018)
39. Zhang, W., Lu, Q., Yu, Q., Li, Z., Liu, Y., Lo, S.K., Chen, S., Xu, X., Zhu, L.: Blockchain-based federated learning for device failure detection in industrial iot. IEEE Internet of Things Journal 8(7), 5926–5937 (2020)
40. Zhou, H., Zheng, Y., Jia, X.: Towards robust and privacy-preserving federated learning in edge computing. Computer Networks 243, 110321 (2024)

Lang Wu Ph.D., is a lecturer at Beijing Information Science and Technology University. She received her Doctor of Science degree from Harbin Institute of Technology in

2016. Her research focuses on the application of artificial intelligence in complex systems, with particular emphasis on meta-learning, federated learning, privacy-preserving computation, and personalized model optimization. She is dedicated to addressing uncertainty modeling and intelligent decision-making in high-dimensional data analysis. Dr. Wu built a solid foundation in mathematical theory through early work on high-precision numerical solutions of partial differential equations. Later, she shifted her focus to the deep integration of machine learning with real-world industrial applications, proposing various optimization methods that have significantly enhanced the practicality and robustness of models in critical fields such as finance and healthcare. In recent years, she has published more than ten papers in domestic and international academic journals, and some of her research outcomes have been successfully applied in real-world systems. She possesses strong research capabilities and a solid foundation in engineering practice.

Yi Dong is a master's student at Beijing Information Science and Technology University, primarily engaged in research on federated learning. Her work focuses on data privacy protection and distributed modeling techniques, aiming to enhance model generalization and robustness in multi-source heterogeneous data environments. Under the guidance of her supervisor, she actively participates in research projects, demonstrates strong programming skills and academic competence, and is exploring the application potential of federated learning in real-world scenarios such as finance and healthcare.

Received: November 19, 2024; Accepted: May 10, 2025.