Computer Science and Information Systems 22(3):945–970 https://doi.org/10.2298/CSIS250131052Z

PFLIC: A Novel Personalized Federated Learning-Based Iterative Clustering

Shiwen Zhang[1,2], Shuang Chen[1,2], Wei Liang[1,2], Kuanching Li[1,2,⋆], Arcangelo Castiglione[3], and Junsong Yuan[4]

[1] School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China
[2] Sanya Research Institute, Hunan University of Science and Technology, Sanya 572024, China
{shiwenzhang,wliang,aliric}@hnust.edu.cn
shuangchen@mail.hnust.edu.cn
[3] Department of Computer Science, University of Salerno, Fisciano, SA, Italy
arcastiglione@unisa.it
[4] University at Buffalo, State University of New York, Buffalo 14201, New York, USA
jsyuan@buffalo.edu

Abstract. Federated learning (FL) is a machine learning framework that effectively helps multiple organizations perform data usage and machine learning models while meeting the requirements of user privacy protection, data security, and government regulations. However, in practical applications, existing federated learning mechanisms face many challenges, including system inefficiency due to data heterogeneity and how to achieve fairness to incentivize clients to participate in federated training. Due to this fact, we propose PFLIC, a novel personalized federated learning based on an iterative clustering algorithm, to estimate clusters to mitigate data heterogeneity and improve the efficiency of FL. It is combined with sparse sharing to facilitate knowledge sharing within the system for personalized federated learning. To ensure fairness, a client selection strategy is proposed to choose relatively "good" clients to achieve fairer federated learning without sacrificing system efficiency. Extensive experiments demonstrate the superior performance and effectiveness of the proposed PFLIC compared to the baseline.

Keywords: Federated learning, Clustering algorithm, Client Selection, Sparse sharing.

1. Introduction

Contemporary mobile smart technologies have reached an unprecedented level of sophistication. This technological evolution enables ubiquitous smartphone/sensor applications [17], which collectively produce real-time data streams at an extraordinary scale [32, 34]. The question of how to utilize and store these data has become a hot topic across industries. In traditional deep learning, these data are collected and stored in a central location to train neural networks. However, in some special cases, the data owner is reluctant to share these data with a third party due to privacy protection. To solve this problem, a promising distributed machine learning method, Federated Learning (FL) is proposed [12]. The

⋆ Corresponding author

traditional framework of FL is illustrated in Fig. 1, which learns a global model that aggregates information from each client while protecting participants' privacy.

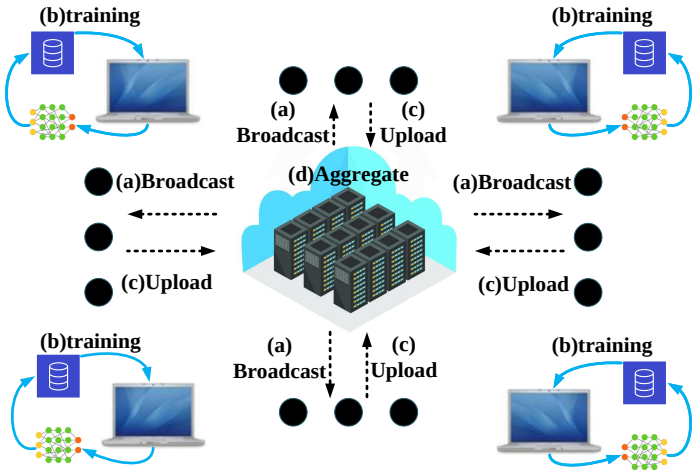


Fig. 1. A general FL framework. (a) Cloud server broadcasts the global model. (b) Clients training model. (c) Clients upload the local model. (d) Cloud server aggregation model.

Although FL can solve a distributed machine learning model without anyone seeing or touching the raw data of each client, several problems need to be addressed for efficiency, especially those regarding data heterogeneity and fairness of FL. In real applications, different clients generate data in various ways, resulting in non-independent and identically distributed (non-iid) data among clients, also known as data heterogeneity [20, 33]. Several studies have found that data heterogeneity can seriously affect the system's convergence and the model's accuracy [19, 20]. On the other hand, it is necessary to ensure FL's fairness by not favoring either party while achieving global knowledge sharing. If fairness cannot be guaranteed, some clients with relatively small contributions, *i.e.*, those less involved in FL training, will terminate their participation at any time. Therefore, how to design an efficient federated learning framework that solves data heterogeneity and protects fairness is of paramount importance.

Data heterogeneity is one of the central issues in the development of FL. To address this problem, many researchers have devoted themselves to designing a number of FL schemes [2, 19, 20, 23, 29]. To mitigate the impact of data heterogeneity, some have used Personalized Federated Learning (PFL) based on clustering algorithm [2,23,29] to cluster clients with similar data distributions and divide them into the same cluster to train a model dedicated to each cluster. Others have penalized local models that deviate too much from the global model to prevent local models from deviating from the global model [12, 15, 16, 31]. In addition, some have processed client data by dividing the dataset or data augmenting to synthesize regular datasets that correct for the heterogeneity of private

data for the client [5, 27]. Nevertheless, the penalty mechanism and data augmentation cannot solve the data heterogeneity problem. On the contrary, the clustering algorithm starts from data distribution, clusters clients with the same data distribution, and trains a cluster model for each cluster instead of a single global model, which solves the problem caused by data heterogeneity. However, the centralized clustering algorithm will bring the issue of high communication overhead, the iterative clustering algorithm can effectively reduce the overhead. The iterative clustering algorithms for PFL in FL systems introduce a new problem, *i.e.*, the inability of global knowledge sharing among groups with different data distributions.

How to incentivize active client participation in training and achieve equity is another central issue in the development of FL. To solve this problem, some pay more attention to the fairness of the results and expect to achieve the equilibrium fairness [9, 10, 25], *i.e.*, the model performs equilibrium among the clients and does not favor a specific data party. However, the robust data side is more likely to be selected in this process, and the final trained global model will also be biased in favor of the firm side, resulting in the data's inferior side being ignored. Other researchers focus more on the fairness of the process and want to achieve contribution fairness [18, 22], *i.e.*, allowing differences in model performance but pursuing balanced contributions from all parties. Nevertheless, this may cause dissatisfaction among high-contributing clients, leading to the problem of "free-riding" [26] and affecting the sustainable development of FL. For the sustainability of FL, we need to be unbiased towards either party, ensure fairness, and allow for the case where models perform differently. Hence, it is still challenging to mitigate the impact of data heterogeneity, following the client's most primitive features and ensuring the system's fairness without increasing the communication overhead.

Unlike previous studies, we design PFLIC, a novel personalized federated learning based on an iterative clustering algorithm in this work, to effectively address the issue of data heterogeneity and achieve the fairness of FL. We leverage the similarity among clients to implement iterative clustering and thus improve the convergence speed of such a system to address data heterogeneity. Besides, we integrate weight sharing in multi-task learning to enable PFL and facilitate inter-cluster collaboration to minimize communication overhead. To maintain the fairness of FL, we design a client selection strategy to select the clients to participate in the training process to guarantee the participation rate of clients in the training process. This work mitigates the data heterogeneity problem while maintaining FL's fairness without sacrificing accuracy. The contributions of this work are listed as follows:

- To solve the problem of data heterogeneity and system unfairness, minimizing system overhead, and enabling personalized federated learning, we propose and design PFLIC.
- To alleviate the heterogeneous data problem, we design an iterative clustering algorithm to cluster customers with high similarity by continuously determining their identities before training. To achieve PFL, we combined the weight sharing to drop to achieve knowledge learning between clusters and reduce the system's communication overhead.
- To ensure the fairness of federated learning, we design a client selection strategy to actively select "good" clients according to the established metrics. This means that

the weak clients are no longer feeble. And the model not only equalizes performance between clients but also allows for variance.
– To demonstrate the performance of PFLIC in terms of accuracy and efficiency, we conduct extensive experiments on real-world datasets. Compared to the baseline, experimental results show that the proposed method achieves promising results.

The remainder of this article is organized as follows. The related work is introduced in Section II, the system model and problem formulation are described in Section III, the construction and workflow of PFLIC are presented in Section IV. Experimental results and analysis are provided to show the superiority of PFLIC over the baseline methods in Section V. Finally, the concluding remarks and future directions are given in Section VI.

2. Related work

2.1. Data Heterogeneity in Federated Learning

A central issue in developing FL systems in recent years is heterogeneity, categorized into data heterogeneity and structural heterogeneity. To solve the problem of poor model performance due to data heterogeneity, several works [2,7,19,23] consider using clustering to address data heterogeneity. Clustering Federated Learning (CFL) is a promising approach to solving the data heterogeneity problem. Sattler *et al.* [19] proposed a methodological framework for federated client grouping learning, an algorithm in which the parameter server dynamically divides the participants based on their gradient or update information. However, the server has a high computational cost. Tu *et al.* [23] dynamically learned personalized models for different users by learning the similarity between user model weights to form a shared structure. Briggs *et al.* [2] combined hierarchical clustering with FL to separate client clusters by calculating the similarity of the client's local updates to the global model. After separation, the clusters are trained independently and in parallel on specialized models. Tu *et al.* [23] and Briggs *et al.* [2] divided the clients into clusters by calculating the distance of the local model weights, which improves the accuracy but brings about slower convergence. Ghosh *et al.* [7] divided similar client data distributions into a cluster, but their clustering results are unstable and have some impact on the model accuracy. Unlike the single clustering algorithm mentioned above, we use an iterative clustering algorithm, which reduces the high communication overhead associated with centralized clustering algorithms.

2.2. Fairness in Federated Learning

Client selection has become an emerging topic that addresses fairness in FL. It chooses which clients will participate in each round of training. If all clients are involved in each round of training, the communication cost will be high. Thus, using a client selection strategy is also an excellent way to reduce the cost of communication. A good selection strategy can improve the model accuracy, reduce the training cost, and enhance the fairness of the system [1, 11, 13, 14, 21, 28]. Tang *et al.* [13], Lai *et al.* [21] and Cho *et al.* [11] improved the convergence speed of the model by implementing the client selection function. Tang *et al.* [13] argued that the clients do not contribute equally and do not contribute independently, so they use the loss correlation of clients for client selection.

Lai *et al.* [21] designed the Oort framework that allows developers to specify on their own what kind of federated learning clients can be added, combining fairness and statistical usage. Cho *et al.* [11] made a trade-off between convergence speed and solution bias and found that biasing client selection toward clients with higher local loss of clients achieves faster error convergence. Xu *et al.* [28] argued that optimizing the learning performance depends critically on how the clients are selected, but it only considers the data heterogeneity. Li *et al.* [14] selected clients to achieve fairer network performance. Li *et al.* [14] and AbdulRahman *et al.* [1] selected clients based on different strategies to improve the global accuracy of the model and the speed of convergence. However, they may destroy the clustering structure.

2.3. Personalized Federated Learning

The strategies for implementing personalized federated learning can be divided into global model personalization and learning customized models. The former intends to enhance the performance of a global model federally trained on heterogeneous data. Wu *et al.* [27] augmented local datasets using a self-encoder, which enhances the usability of the local dataset to represent the overall data distribution, but with the possibility of privacy leakage. Wang *et al.* [24] used deep learning for training to select the participating clients to mitigate the effect of non-iid data. This approach samples from a more homogeneous data distribution, improving model generalization performance even though it may incur higher computational costs. The latter is intended to provide personalized solutions by modifying the FL model aggregation process to build customized models. Bui *et al.* [3] considered personalized feature representations for each client by using users as private model parameters but are limited in supporting a high degree of model design personalization. Annavaram *et al.* [8] proposed population knowledge transfer through a bidirectional distillation approach using alternating minimization to train local and global models to support personalized model architectures for each client, but this can lead to inferior training of student models if there are too many differences between the teacher model and the student model. With the help of sparse sharing techniques, we allow knowledge learning between different clusters to achieve personalized federated learning while guaranteeing the accuracy of models within this cluster.

3. Overview of PFLIC

This section presents the system model, the problem formulation, and the description of the design goal, while Table 1 summarizes notations commonly used throughout this work.

3.1. System Model

PFLIC is a novel personalized federated learning scheme that can solve the problem of non-iid data, improve the model's accuracy and convergence speed of the system, and reduce communication costs. Specifically, we propose clustering before training to solve the excessive difference in data distribution of each user in the FL system. The process is iterated during training to prevent incorrect identity estimation at the first clustering. On this

Table 1. Summary of Notations

Notation	Explanation
$\hat{\theta_t^j}$	Model of client j in round t
θ_t^i	Cluster model i in round t
N	Number of clients
k	Number of clusters
c_i	One cluster i in the set of all clusters
P_t	Clients selects for training at round t
D	Total quantity of data
D_j	Data of client j
id_j	The identity of client j
$\mathcal{L}_j$	Loss function of client j
C_j	The utility value of client j
S_j	The total number of training rounds for a client
a_j	Accuracy of client j
η	Learning rate
T	Training rounds
TS	Threshold for client participation in training
V	Model accuracy distribution variance
A	Global model mean test accuracy

basis, we propose to combine sparse sharing to reduce communication costs and improve convergence. Finally, we design a client selection strategy to actively select clients participating in training to achieve fairer federated learning. Fig. 2 shows the overall system model of PFLIC, which can be categorized into two cases:

The first case is when the clustering results are not stable. Firstly, the cloud server broadcasts k cluster models to the client; then the client uses the received models and local data to estimate the cluster identity, uses the local data to update the models, and uploads the trained models to the cloud server; secondly, the server determines whether the clustering results are stable or not, and aggregates the received models within each cluster.

The second scenario is that the clustering results are already stable. Firstly, the cloud server broadcasts the weight subsets and shared layers of the models of the clusters of the class to the clients; then the clients use local data to update the models and upload the trained models to the cloud server; finally, the cloud server performs client selection, as well as aggregation of the received models within each cluster respectively.

More precisely, the whole process of the program involves two interactions between the server and the client. The first iteration is used for clustering and the second iteration is used for active client selection and model updating.

3.2. Problem Formulation

One center cloud server and N clients exist in personalized federated learning settings. The cloud server and clients can communicate using a predefined communication protocol. Clients have different data, denoted as $\{D_1, D_2,...,D_N\}$. In this work, we assume

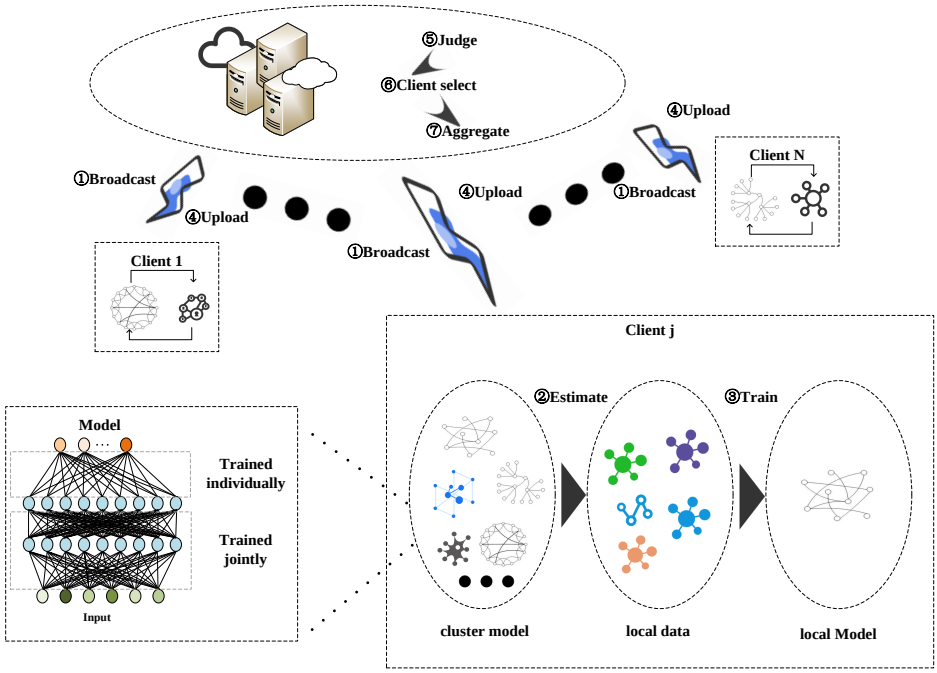


Fig. 2. System model of PFLIC

that there are potential clustering relationships between clients' data, which can be divided into k clusters, denoted as $\{ c_1, c_2,...,c_k \}$, whereas the goal is to learn k good cluster models θ^{i*}:

$$\theta^{i*} = argmin_{j\in[k]}\mathcal{L}_j(\theta_i), i \in [k]\,. \tag{1}$$

For each cluster by combining information from all cluster classes without data exchange, where $\mathcal{L}()$ is a loss function.

3.3. Design Goal

The scheme not only effectively solves the data heterogeneity problem and achieves personalized federated learning but also ensures the fairness of FL. Specifically, the FL scheme we designed needs to satisfy the following design goals:

Accuracy: In the absence of other contingencies, the scheme cannot sacrifice the accuracy of the global model. Compared with the baseline, the accuracy of our proposed scheme should be consistent or better.

Efficiency: Due to the limited computational resources of the edge devices, it cannot generate too much extra computational overhead and communication overhead. Compared with the baseline, we propose that the scheme should not add too much workload to the participants.

Fairness: Since the clients participating in the training are self-interested and differ from each other in terms of computational communication resources, data, and others. The sustainability of federated learning needs to maximize client incentives, distribute rewards appropriately, and promote motivation among federated participants. In other words, we need to ensure a certain level of fairness in the system.

4. Design of PFLIC

4.1. Workflow of PFLIC

When the clustering results are not stabilised, the first stage in Fig. 3 is performed. The cloud server distributes k cluster models (line 4). After receiving the cluster model, the server estimates the identity. After the server estimates the cluster identity, it trains using that cluster model and local private data (lines $7-9$). After training, the client uploads the cluster identity and the updated model to the cloud server (line 10). The cloud server aggregates the cluster model separately (lines $26-28$).

When the clustering results are stable, the second phase in Fig. 3 is performed. The cloud server distributes to each participating client the weight subset of the cluster model to which the client belongs and the shared layer (line 18). The client is trained using this model and local data (line 21). After training, the client uploads the trained model parameters to the cloud server (line 16). The cloud server calculates and ranks the utility values of the client and uses the ranked values for client selection(Line 17). The cloud server aggregates the clustered models respectively and uses these aggregated models to generate the shared layer (Lines $27-28$).

4.2. Client Side

FL system is trained jointly on clients having different datasets, where each client dataset has different samples and different kinds of features. Direct model aggregation for models trained with non-iid user datasets affects the model's overall performance, slowing convergence. Thus, we adopt an essential assumption that there are potential clustering relationships between the data of individual clients involved in training. Our goal is to utilize the similarity (gradient) between the client data samples to cluster the clients with higher similarity for training to improve the model's convergence speed and model accuracy for this system.

Since one-shot clustering is prone to chance errors, and once a wrong clustering estimate is generated, it cannot be corrected in any subsequent training phase, it will impact the whole FL system training. Therefore, iterative clustering is used in this paper. Before the clustering results are stabilized, the cloud server performs a clustering analysis before aggregating the models. While training during training, the clustering results are dynamically adjusted according to the model parameters $\widetilde{\theta}_t$. Fig. 4 compares primary and iterative clustering.

Algorithm 2 gives pseudo-code for iterative clustering. The global model obtained in this step depends on the clustering and client selection results. In the t-th training round where the clustering results are not stabilized, all clients receive models $\theta_t^i (i \in [k])$

Algorithm 1 PFLIC

Input: initialize parameters $\theta_0^i (i \in [k])$, number of all clients N, learning rate η, number of clusters K

for all $t = 0, 1, ..., T$ **do**
  SERVER SIDE:
  if the clustering results are not stabilized **then**
    Broadcast k models $\theta_t^i (i \in [k])$
    CLIENT SIDE:
    for all each client $i \in [k]$ **do**
      Compute $\hat{i} = argmin_{i \in [k]} \mathcal{L}_j(\theta_t^i, D_j)$
      Estimated $id_j = \{id_{i,j}\}_{i=1}^{k}$, $id_{i,j} = 1\{i = \hat{i}\}$
      Compute $\hat{\theta_t^j} = \theta_t^i - \eta \nabla \mathcal{L}_j(\theta_t^i, D_j)$
      Send $\hat{\theta_t^j}$,id_j to server
    end for
    SERVER SIDE:
    Cluster $\{\hat{\theta_t^j}\}_{i=1}^{N}$ into $c_1, c_2, ...c_k$
  else
    SERVER SIDE:
    Clients selection using Algorithm 3
    $P_t = participatingclients$
    Broadcast one shared layer and k subsets of different versions of weights
    CLIENT SIDE:
    for all each client $i \in [k]$ **do**
      Compute $\hat{\theta_t^j} = \theta_t^i - \eta \nabla \mathcal{L}_j(\theta_t^i, D_j)$
      Send $\hat{\theta_t^j}$ to server
    end for
  end if
  SERVER SIDE:
  for all each cluster$(c_1, c_2, ...c_k)$ in parallel **do**
    $\theta_{t+1}^i = \theta_t^i + \sum_{j=1}^{N} \frac{D_j}{D} \hat{\theta_t^j}, i \in [k]$
    Cloud server generates shared layers using cluster model
  end for
end for

Algorithm 2

Input: number of clients N, loss function $\mathcal{L}$, number of clusters k, clients P_t participating in training at round t

Server broadcast $\theta_t^i, i \in [k]$
Clients($j \in P_t$) for identity estimation and training
Clients($j \in P_t$) send model $\hat{\theta_t^j}$ and id_j to the CS
for all each each cluster$(c_1, c_2, ..., c_k)$ in parallel do **do**
  $\theta_{t+1}^i = \theta_t^i + \sum_{j=1}^{N} \frac{D_j}{D} \hat{\theta_t^j}, i \in [k]$
end for

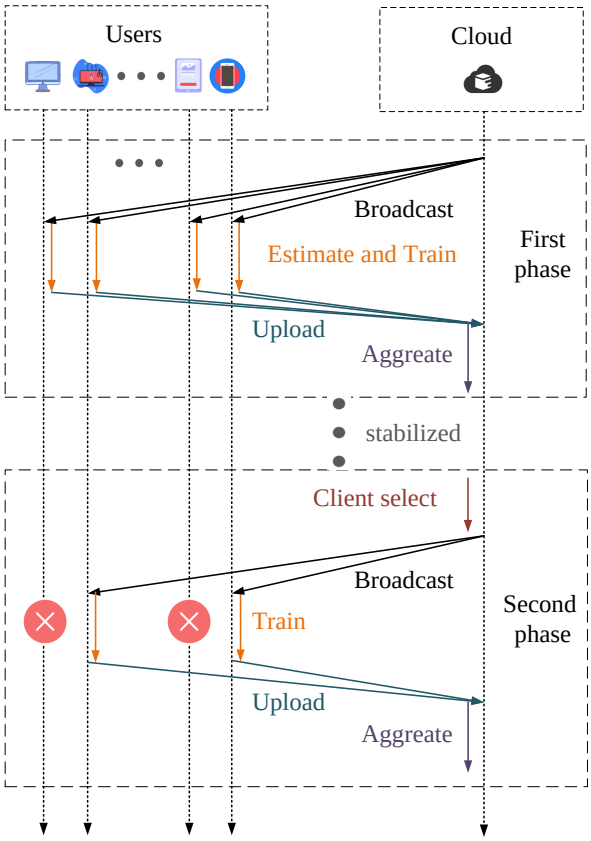


Fig. 3. Workflow of PFLIC

broadcast from the cloud server and use these models and its local empirical loss function $\mathcal{L}_j()$ to find the model parameter that minimizes loss by $\hat{i}$:

$$\hat{i} = argmin_{i\in[k]}\mathcal{L}_j(\theta_t^i, D_j), j \in [N]\,. \tag{2}$$

The identity id_j:

$$id_j = \{id_{i,j}\}_{i=1}^{k}\,. \tag{3}$$

$id_{i,j} = 1\{i = \hat{i}\}$ of this client is determined by using $\hat{i}$ to estimate which cluster this client is in after clustering. Then use $\theta_t^{\hat{i}}$ to perform stochastic gradient descent training by using local data to compute the model parameter $\hat{\theta_t^j}$ of $\mathcal{L}_j$. Therefore, the client sends the model parameter result $\hat{\theta_t^j}$ and the clustering identity id_j to the cloud server. When the clustering results are stabilized, all clients involved in the training receive the model $\theta_t^{\hat{i}}$ broadcast from a cloud server, train it by local stochastic gradient descent, and update the model. When a predetermined number of local training sessions is reached, the client uploads the parameters to a cloud server.

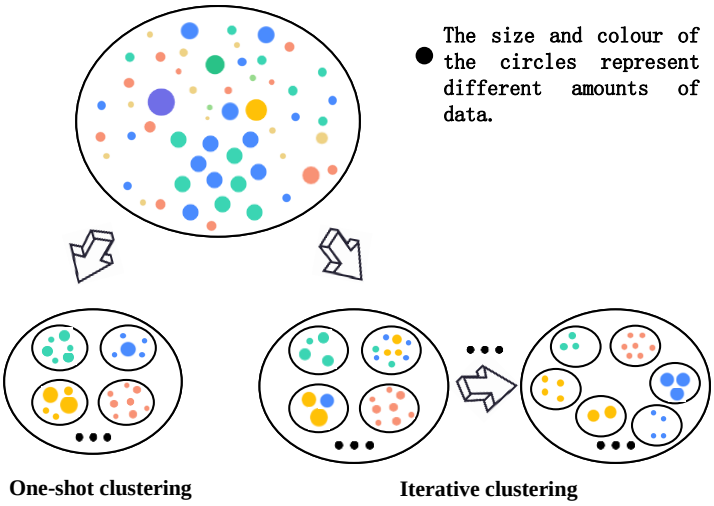


Fig. 4. One-shot clustering vs. Iterative clustering. *Left*: When one-shot clustering is performed, the client's participation in the training does not represent the client's overall data distribution, resulting in an incompletely accurate estimate of the client's identity. *Right*: When iterative clustering is performed, it is trained several times before clustering is performed.

4.3. Server Side: Broadcast

Compared to centralized machine learning, where computational costs dominate and communication costs are negligible, communication costs in FL are much higher than computational costs. Based on previous experience, the communication cost is directly related to the parameters transmitted among participants. To solve the problem of high communication costs in FL, we draw on the sparse sharing in sparse sharing proposed by SUN *et al.* [30] to allow the sharing of some parameters among different clusters to achieve PF and reduce the communication cost. Fig. 5 utilizes two representations of the sparse sharing mechanism to illustrate the concept of sparse sharing.

To reduce the communication cost, we combine sparse sharing by replacing k models broadcast by the cloud server with one shared layer and k subsets of different versions of weights, which reduces the transmitted model parameters and lowers the communication cost. Specifically, At the first FL system training round, the cloud server initializes k models $\theta_0^i (i \in [k])$. After that, these models are sent to all clients $j (j \in [N])$. In the t-th round of FL training where the clustering results are not stabilized, the cloud server sends k subsets of the models $\theta_t^i (i \in [k])$ and a shared layer to all clients. When the clustering results are stabilized, the cloud server sends a weighted subset of the model $\theta_t^i (i \in [k])$ with corresponding clusters and a shared layer to all clients selected to participate in the training.

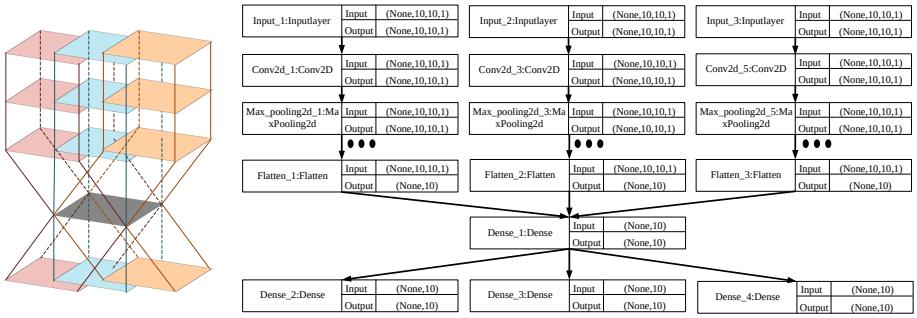


Fig. 5. Two representations of the same layered sharing mechanism case (Three models share a single layer). *Left*: Expressed using a convolutional neural network graph structure. *Right*: Expressed using a function called 'building block' form

We use sparse sharing, which not only reduces the number of transmitted parameters and lowers the communication cost; it also allows the sharing of task parameters between different clusters, breaks down the barriers between different clusters, facilitates the sharing of knowledge between clusters, and improves the convergence speed and accuracy of the system.

4.4. Server Side: Client Select

After receiving the model $\theta_t^i (i \in k)$, the client estimates its clustering identity through training using its experience loss function $\mathcal{L}()$. After receiving the uploaded identity estimation, the cloud server first determines whether the clustering result is stable or not. If it is not stable, the clustering continues. If it has been stable, no further clustering operation is performed for subsequent training, and client selection begins.

The previous client selection scheme is generally random [20], which leads to some clients with unique data distributions being challenging to select; there is little variability in the clients selected by extraction; some clients are extracted frequently, etc., which reduces the representativeness of the client population and makes the convergence of the global model more unstable. This limitation may affect the clustering results. To solve this problem, this paper proposes a client selection strategy, which selects relatively "good" clients to achieve fairer federated learning. Algorithm 3 outlines pseudo-code for this part.

To solve the problem raised above and improve the fairness of FL, this strategy selects clients with higher losses and considers their participation rounds simultaneously. Specifically, after the client uploads the parameters to a cloud server, the server calculates the value $\mathcal{C}$ of the client based on the received loss. Then, the server prioritizes all clients based on this value $\mathcal{C}$. Assuming that the total number of training rounds for a client in round t is S_j, we define this value $\mathcal{C}$:

$$\mathcal{C} = \frac{\sum_t^{S_j} \mathcal{L}_{j_t}}{S_j} . \tag{4}$$

Algorithm 3 Client Select

Input: number of clients N, loss function $\mathcal{L}$, number of clusters k, clients P_t participating in training at round t, participation rounds t_j for the t-th client, value of utility measures C_t^j, threshold TS for client participation in training

1: **while** $t_j < TS$ **do**
2: Add 1 to the number of rounds t_j for client j participating in the training
3: Clients P_t involved in training utilize local data for training
4: Compute the value of utility measures of the client j: $C_j = \sum_t^{S_j} \mathcal{L}_{j_t}$
5: Sorting the client's value of utility measures
6: Select the top n clients with the largest value C in each cluster for the next round of training
7: **end while**

After sorting, clients with larger value $\mathcal{C}$ will be selected to participate in training. According to the above equation, it can be concluded that clients with larger loss values have a greater chance to participate in training. This aspect makes the model accuracy distribution variance smaller, the client model accuracy distribution more balanced, and the FL system more fair. $\mathcal{V}$ is defined as

$$\mathcal{V} = \frac{\sum_{j=1}^{N}(a_j - \mathcal{A})^2}{N}. \tag{5}$$

Where $\mathcal{A} = \frac{\sum_{j=1}^{N} a_j}{N}$ is the global model average test accuracy, a_j is the accuracy of each participant, and N is the total number of clients. To ensure training efficiency, a participation threshold mechanism is implemented, limiting the maximum number of client engagements per training round. When a client's training rounds exceed this threshold, the client will no longer participate in training, and it would increase the participation rate of other clients.

The client selection strategy proposed in this section actively selects clients that participate in training, which reduces the variance of the model accuracy distribution and improves the accuracy of both the client and the global model; setting a threshold prevents clients from endlessly participating in training and enhances the participation rate of other clients.

4.5. Efficiency Analysis

The convergence of iterative clustering in FL has been demonstrated in previous studies [7]. Furthermore, Cho *et al.* [4] showed that a biased client selection strategy does not affect the convergence properties of FL. Therefore, a biased client selection framework also does not change the convergence property of CFL. Thus, we focus on evaluating the efficiency of our proposed PFLIC and baseline algorithms (FedAvg and CFL). The definitions used for our analysis are provided next.

Definition 1: Efficiency(E) is defined as the sum of the computational efficiency(E_{Cal}) and the communication efficiency(E_{Com}). Therefore, the efficiency can be written as follows: $E = E_{Cal} + E_{Com}$.

Definition 2: Computational efficiency(E_{Cal}) is defined as the total computational cost required to train the model to achieve the desired test accuracy threshold. Assuming

that the expected test accuracy threshold is Acc, the corresponding number of training rounds spent by the algorithm is denoted $round$. Additionally, the computational cost of one iteration of the algorithm is Cal. Therefore, the computational efficiency E_{Cal} can be written as follows: $E_{Cal} = Cal * round$.

Definition 3: Communication efficiency(E_{Com}) is defined as the total communication cost required to train the model to achieve the expected test accuracy threshold. Assuming that the expected test accuracy threshold is Acc, the corresponding number of training rounds spent by the algorithm is denoted as round. Furthermore, the communication cost of one iteration of the algorithm is Com. Therefore, the communication efficiency E_{Com} can be written as follows: $E_{Cal} = Cal * round$.

Based on the number of clients N, the number of clusters k, the participation rate ρ, the number of model parameters per participant P, and the number of training rounds $round$, we present the results of the computational efficiency for different algorithms in Table 2.

Table 2. Results on computational efficiency between different algorithms

Scheme	E_{Cal}
FL	$E_{Cal}^{FL} = N * Cal * round$
CFL	$E_{Cal}^{CFL} = N * Cal * round + N * k * log(N)$
PFLIC	$E_{Cal}^{PFLIC} = N * (Cal + k) * t + N * \rho * Cal * (round - t)$

Computational efficiency: Each participant in FL performs local training, so the computational complexity of each round is the number of participants N multiplied by the training cost Cal of each participant. Therefore, the computational efficiency E_{Com} of FL can be written as follows:

$$E_{Cal}^{FL} = N * Cal\,. \tag{6}$$

Clustering federated learning requires clustering operations in addition to the complexity of local training. Assuming that the algorithm complexity used for clustering is $O(N * log(N))$, then the computational efficiency E_{Cal} of CFL can be written as follows:

$$E_{Cal}^{CFL} = N * Cal + N * k * log(N)\,. \tag{7}$$

The computational overhead of PFLIC is divided into pre-stabilization and post-stabilization computational overheads. Before stabilization, each participant in PFLIC performs local training and identity estimation, then the computational complexity is ${E_{Cal}^{PFLIC}}_{pre} = N * (Cal + k)$. After stabilization, PFLIC performs client selection, and the selected participant performs local training, then the computational complexity is ${E_{Cal}^{PFLIC}}_{post} = N * \rho * Cal$.

Communication efficiency: Federation learning requires each participant to send model parameters to the central server after each iteration round, so the communication complexity of FL can be written as follows:

$$E_{Com}^{FL} = N * P\,. \tag{8}$$

Clustering federation learning after clustering, only the clustering center communicates with the central server, so the communication complexity of CFL can be written as follows:

$$E_{Com}^{CFL} = K * P\,. \tag{9}$$

The communication overhead of PFLIC is divided into pre-stabilization and post-stabilization communication overhead. Before stabilization, each participant needs to send model parameters to the central server, so the communication complexity is $E_{Cal}^{PFLIC}{}_{pre} = N * P$. After stabilization, only the selected participants need to send model parameters to the central server, so the communication complexity is $E_{Com}^{PFLIC}{}_{post} = N * \rho * P$, we present the results of the communication efficiency for different algorithms in Table 3.

Table 3. Results on communication efficiency between different algorithms

Scheme	E_{Com}
FL	$E_{Com}^{FL} = N * P * round$
CFL	$E_{Com}^{CFL} = K * P * round$
PFLIC	$E_{Cal}^{PFLIC} = N * P * t + N * \rho * P * (round - t)$

5. Experimental Results

5.1. Experiment Settings

Models and Datasets: We conducted experiments on two real datasets, MNIST and CIFAR-10). To adhere to the assumption of a potential clustering relationship among cross-client data, we refer to Ghosh *et al.* [7] for rotating data on the MNSIT dataset. In MNIST and CIFAR-10 experiments, We used two Fully Connected Neural Networks (FCNN) and one Convolutional Neural Network (CNN) model that includes two convolutional layers followed by two fully connected layers. In the first FCNN model, we created two fully connected layers and chose to share the last fully connected layer. In the second FCNN model, we created three fully connected layers and chose to share the last fully connected layer. In the CNN model, we share one convolutional layer and utilize it to demonstrate the general applicability of our proposed algorithm.

Benchmarks: We compare the performance evaluation of our proposed algorithm with three well-known federated learning algorithms. The first comparison scheme is the Fedavg algorithm [20] with improved communication overhead, which reduces the communication overhead of the system by reducing the number of communication rounds in the federated learning process compared to the previous algorithms. We also compared it with a one-shot clustering algorithm [6]. [6] provides two different clustering algorithms based on sample size(one-shot-1) and model similarity(one-shot-2), which do not require any additional operations on the client side and can be seamlessly integrated into standard FL implementations. The fourth benchmark [7] is an iterative federated clustering algorithm that alternately estimates the clustering identities of users and optimizes the model parameters for user clustering via gradient descent.

Performance Metrics: We will focus on the loss value, the accuracy, the convergence rate, and the number of communication rounds. The effectiveness of the scheme is verified by the first three metrics, and the overhead of the scheme is illustrated by the last metric.

Experiment Parameters: In all experiments, we default the learning rate θ is 0.01. We default the number of local updates per epoch to $H = 10$. In the MNIST and CIFAR-10 datasets, we randomly distributed the data evenly across all clients. To avoid accidents, we used the average results of multiple independent experiments in all experiments.

5.2. Effects of the proposed scheme

Effect of Clients Number: To determine the impact of the amount of data owned by the user on our scheme, after fixing the number of clusters, we use the same dataset and set a different number of clients, then the data samples assigned to each client is also changed. The number of clients varies, and the amount of data the clients have also varies; thus, we determine whether we can affect the whole system's performance.

We cannot arbitrarily set the number of clients due to the effect of the sample size of the dataset itself. Thus, in the MNIST dataset, we set the number of clients m in the training set to 48, 96, and 192 and the corresponding number of clients in the test set to 8, 16, and 32. In the Cifar-10 dataset, we set the number of clients m in the training set to 50, 100, and 200 and the corresponding number of clients in the test set to 10, 20, and 40. In Fig. 6 (a), (b) and (e), (f), we compare the accuracy and loss values of our scheme for three different numbers of users. In Fig. 6 (c), (d) and (g), (h), we compare the standard deviation of the accuracy and the standard deviation of the loss values of our scheme for three different numbers of users. As shown in Fig. 6, the performance of m = 48, 96, and 192 is very close. By the time the run reaches 40 rounds, it has roughly stabilized with an accuracy of 97% and a loss below 0.1.

Intuitively, the size of the data volume owned by the client slightly affects the performance of our scheme, which was demonstrated experimentally when the proposed scheme converges and stabilizes to a sure accuracy after a certain number of training rounds.

Accuracy and Convergence: We designed two sets of experiments in which the performance metrics compared were accuracy and convergence speed. One set was used to compare the overall performance of PFLIC with the other three baseline methods, and one set used empirical loss to compare model robustness. We compare the model performance of each scheme for a certain number of communication rounds in Fig. 7 and 8, respectively.

In the MNIST dataset, it is observed that the iterative clustering idea plays a vital role in improving the performance of the federated learning system, both for training CNN models and FCNN models. In terms of accuracy, when training the CNN model, as shown in (a) in Fig. 7, our scheme and IFCA reach 99.40$\%$ after stabilization, while the one-shot clustering scheme and the traditional federated learning scheme are below 80%. When training the FCNN model, as shown in Fig. 7 (b) and (c), our scheme PFLIC is also more accurate than the clustered federated learning and traditional federated learning schemes. Regarding convergence speed, PFLIC and IFCA schemes with iterative clustering ideas are far better than the single and traditional federated learning schemes. Our scheme has converged and stabilized in the fiftieth round, while the conventional federated learning and one-shot clustering schemes are still fluctuating. Due to the data distribution imbal-

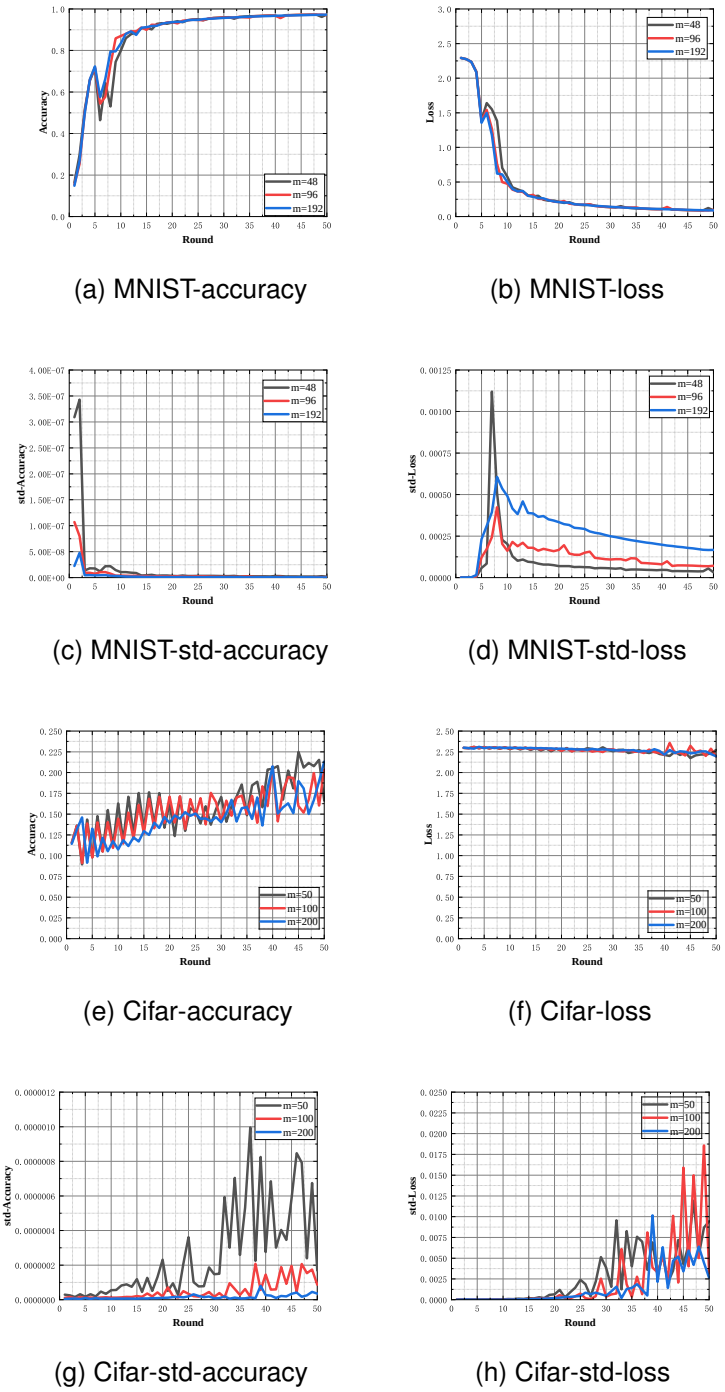
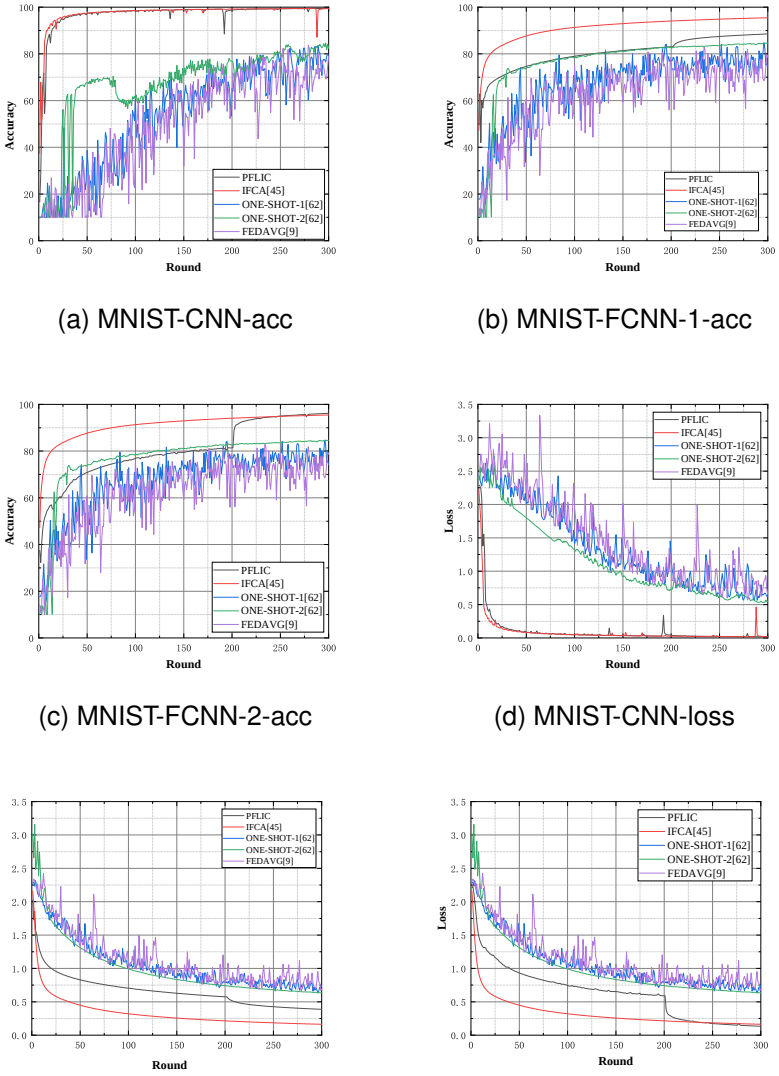


Fig. 6. Comparison of PFLIC schemes under different number of clients (m = 48, 96, 192). (a), (b), (c), and (d) are compared in the MNIST dataset. (e), (f), (g) and (h) are compared in the Cifar-10 dataset. Specifically, (a) and (e) compare the accuracy of our scheme for three different numbers of clients under different datasets, respectively. (b) and (f) Compare the loss of our scheme for three different numbers of clients under different datasets, respectively. (c) and (g) Compare the standard deviation of the accuracy of our scheme for three different numbers of clients under different datasets, respectively. (d) and (h) Compare the standard deviation of the loss of our scheme for three different numbers of clients under different datasets, respectively

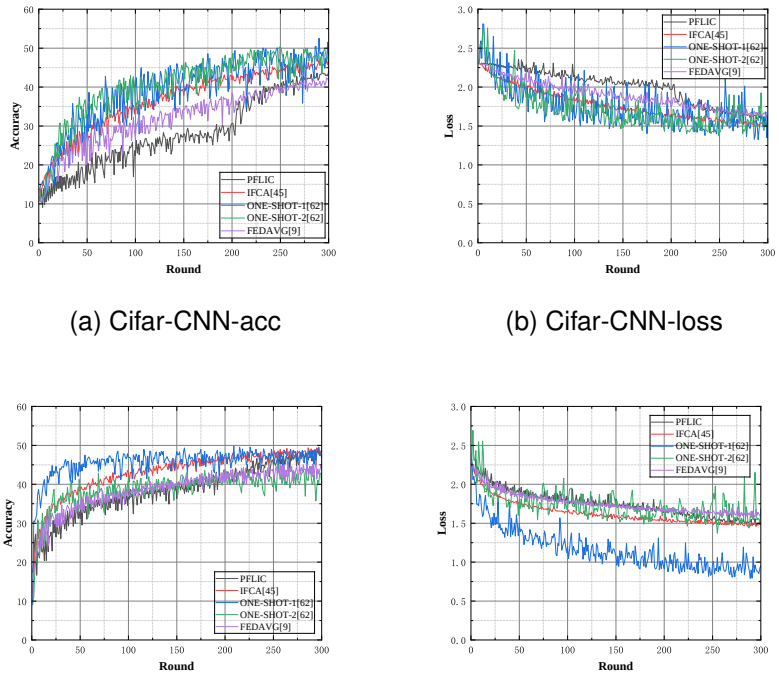


(a) MNIST-CNN-acc (b) MNIST-FCNN-1-acc

(c) MNIST-FCNN-2-acc (d) MNIST-CNN-loss

(e) MNIST-FCNN-1-loss (f) MNIST-FCNN-2-loss

Fig. 7. Comparison of our scheme PFLIC with FL, one-shot CFL, and IFCA regarding scheme performance. (a), (b), (c), (d), (e), and (f) compare the three schemes using CNN models and FCNN models under the MNIST dataset, respectively. Specifically, (a) illustrates the accuracy of training CNN models. (b) Illustrates the accuracy of training the first class of FCNN models (one layer shared and one layer trained separately). (c) Illustrates the accuracy of training the second class of FCNN models (one-layer shared and two layers trained separately). (d) Illustrates the loss of training CNN models. (e) Illustrates the loss of training the first class of FCNN models (one layer shared and one layer trained separately). (f) Illustrates the loss of training the second class of FCNN models (one layer shared and two layers trained separately)



(a) Cifar-CNN-acc (b) Cifar-CNN-loss

(c) Cifar-FCNN-acc (d) Cifar-FCNN-loss

Fig. 8. Comparison of our scheme PFLIC with FL, one-time CFL, and IFCA regarding scheme performance. (a), (b), (c), and (d) compare the three schemes using CNN models and FCNN models under the Cifar-10 dataset, respectively. Specifically, (a) illustrates the accuracy of training CNN models. (b) Illustrates the loss of training the CNN models. (c) Illustrates the accuracy of training FCNN models. (d) Illustrates the loss of training the FCNN models

ance in the client, the model briefly fluctuates after it stabilizes. Our scheme PFLIC shows fluctuation in around 190 rounds, while IFCA shows it in around 280 rounds.

In the Cifar-10 dataset, our scheme is less prominent in accuracy and convergence performance than before because it does not abide by the assumption that there is a potential cluster relationship between clients to set up the data in advance as the MNIST dataset does. When training the two models, it can be seen from Fig. 8 that after multiple rounds of training, our scheme still has some improvement in accuracy over the traditional federated learning scheme.

When constructing the model, we share some layers to promote knowledge sharing within clusters. This theoretically abandons the model's accuracy and the system's convergence speed to a certain extent, and the model's performance is slightly inferior to that of the IFCA scheme in specific implementations. However, in some specifics, our scheme PFLIC is pretty close to IFCA.

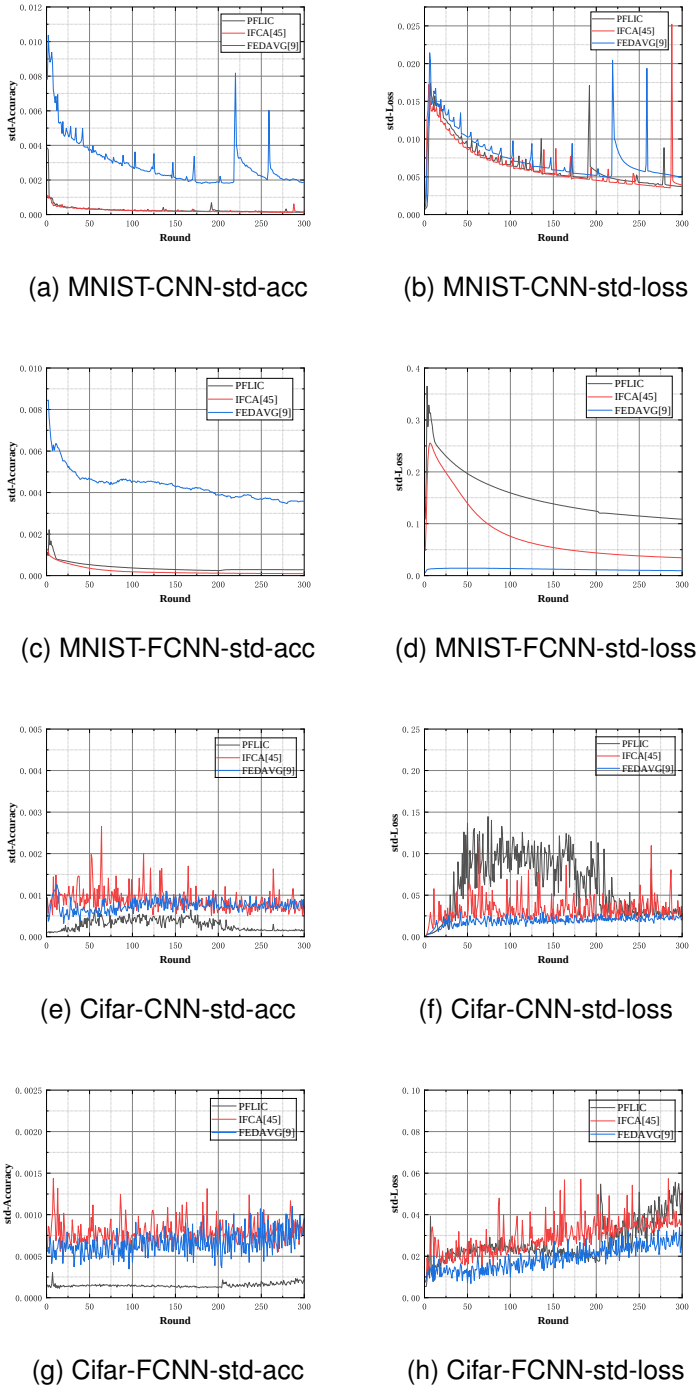
Overhead: We divide the overhead into communication and computation overhead. In this work, the communication overhead refers to the total amount of data and the number of communication rounds required to be transmitted for federated learning to reach a predefined performance metric (e.g., a specific accuracy value).

Regarding the number of communication rounds, we list the comparison results of the number of communication rounds required for different schemes to train the model to reach a specific accuracy for the first time under different datasets in Table 4. It can be seen that the two schemes, PFLIC and IFCA, which possess the idea of iterative clustering, outperform the other three schemes in the experimental results in the MNIST dataset that follows the assumptions. In the Cifar-10 dataset, which does not follow the assumptions, the clustered federated learning scheme (ont-shot-1 and one-shot-2) slightly outperforms the other three schemes. However, our scheme does not lag behind the traditional federated learning scheme either.

Table 4. Comparison of the number of communication rounds required to train a model to achieve a specific accuracy for the first time under different datasets for different schemes

	MNIST									**Cifar-10**					
	CNN			FCNN-1			FCNN-2			CNN			FCNN		
	30%	50%	80%	30%	50%	80%	30%	50%	80%	20%	30%	40%	20%	30%	40%
FL [20]	46	92	279	10	31	176	10	34	176	15	71	245	4	10	123
one-shot-1 [6]	26	92	259	10	18	121	5	26	132	13	41	69	2	3	12
one-shot-2 [6]	24	27	244	5	12	132	10	15	121	16	22	81	4	6	41
IFCA [7]	1	2	8	1	2	13	1	2	13	13	55	143	3	13	58
PFLIC	3	3	10	1	2	119	1	8	157	41	186	233	4	24	130

Regarding the total amount of data transmitted, traditional federated learning only needs to transmit one global model during each communication. Cluster federated learning must only transmit one identity-compliant cluster model to the client during each communication. The IFCA scheme transmits k cluster models during each round of communication. Our scheme PFLIC replaces the k models broadcast by the cloud server with a shared layer and k subsets of different version weights after the clustering is stabilized.



(a) MNIST-CNN-std-acc (b) MNIST-CNN-std-loss

(c) MNIST-FCNN-std-acc (d) MNIST-FCNN-std-loss

(e) Cifar-CNN-std-acc (f) Cifar-CNN-std-loss

(g) Cifar-FCNN-std-acc (h) Cifar-FCNN-std-loss

Fig. 9. Comparison of our scheme PFLIC with FL and IFCA regarding scheme performance. Specifically, (a) and (e) illustrate the standard deviation of the accuracy of training CNN models under different datasets. (b) and (f) Illustrate the standard deviation of the loss of training CNN models under different datasets. (c) and (g) Illustrate the standard deviation of the accuracy of training the first class of FCNN models (one layer shared and one layer trained separately) under different datasets. (d) and (h) Illustrate the standard deviation of the loss of training the first class of FCNN models (one layer shared and one layer trained separately) under different datasets

After the clustering is stabilized, our scheme only needs to transmit a specific model to the client. In FL systems, frequent data transmission inevitably brings about a rise in communication overhead. Compared to the communication overhead, the computational overhead is relatively small and will not be discussed in this paper.

Facts and theories show that although our scheme adds some overheads for knowledge sharing within the system compared to other schemes, the impact is irrelevant to the overall overheads.

Fairness: Since the clients involved in training are self-interested and differ from each other in terms of computational communication resources and data, among others, how to maximize client incentives, rationally distribute rewards, and promote motivation among federated participants is essential for sustainable federated learning. We utilize the standard deviation of accuracy and loss values to illustrate the fairness of our scheme.

Our scheme, PFLIC, selects clients with higher losses and simultaneously considers their number of participating rounds. Fig. 9 shows the results of the compared schemes trained with different models and different datasets. We compare the IFCA scheme, which also has the idea of iterative clustering, with the traditional federated learning scheme. The MNIST dataset that follows the hypothesis shows that the standard deviation of the model accuracy distribution of the two schemes with the iterative clustering idea is lower than the traditional federated learning scheme, indicating a more balanced distribution of client-side model accuracies. It can be seen on the Cifar-10 dataset that our scheme works better than the other two schemes that randomly select clients to participate in training.

Our scheme, PFLIC, actively selects clients to participate in training, which reduces the model accuracy distribution variance and makes the whole system more fair.

5.3. Summary of Experiment Results

From the above experimental results, the iterative clustering idea can significantly accelerate the convergence speed of the system and improve the accuracy of the model to some extent. It significantly outperforms the baseline scheme in both MNIST and Cifar-10 datasets, verifying the versatility of its algorithm design. PFLIC improves the performance and fairness of both CNN and FCNN structures, but its effect varies depending on the model characteristics (CNN improvement is more significant). However, it will increase the overhead of the whole federated learning system when there is no potential clustering relationship between clients.

The client selection strategy can to some extent can improve the fairness of the system, motivate the clients to participate in the training, and promote the enthusiasm of the federated learning participants. The knowledge sharing idea improves the convergence speed and accuracy of the system to a certain extent while allowing different clusters to share task parameters, breaking down the barriers between different clusters, and promoting knowledge sharing between clusters.

Through the synergy of client selection, robust aggregation and communication optimisation, the three core problems of data heterogeneity, communication bottleneck and fairness imbalance in federated learning are solved. All in all, our scheme solves the problem that a single global model cannot adapt to clients with different data distributions, and achieves Personalized federated learning, while ensuring the fairness of the federated learning system.

6. Concluding Remarks and Future Work

In this work, we design and implement a PFLIC scheme to accomplish a novel personalized federated learning. The proposed scheme designed an iterative clustering algorithm that utilizes the similarity among clients to solve the problem of data heterogeneity. It eliminates the chance of single clustering and reduces the computational overhead of single clustering. Subsequently, we combined sparse sharing to facilitate knowledge sharing among clusters and enable personalized federated learning. Moreover, we designed a client selection strategy to ensure the fairness of federated learning. By selecting the clients to participate in the training to ensure the participation rate of all clients in the training process, we prevent some clients from participating in the training process too frequently while others do not have the opportunity to receive training. Experimental results depict that the proposed algorithm performs better than the baseline.

In future directions, we will continue to explore more and better metrics for client selection and ways to deal with clients who drop out of the network due to unstable network conditions.

Acknowledgments. This work is supported by the Natural Science Foundation of Fujian Province under Grant 2022J05106, and the Natural Science Foundation of Hunan Province under Grant 2025JJ50399.

References

1. Abdulrahman, S., Tout, H., Mourad, A., Talhi, C.: Fedmccs: Multicriteria client selection model for optimal iot federated learning. IEEE Internet of Things Journal 8(6), 4723–4735 (2021)
2. Briggs, C., Fan, Z., Andras, P.: Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–9 (2020)
3. Bui, D., Malik, K., Goetz, J., Liu, H., Moon, S., Kumar, A., Shin, K.G.: Federated user representation learning (2019), https://arxiv.org/abs/1909.12535
4. Cho, Y.J., Wang, J., Joshi, G.: Client selection in federated learning: Convergence analysis and power-of-choice selection strategies (2020), https://arxiv.org/abs/2010.01243
5. Duan, M., Liu, D., Chen, X., Liu, R., Tan, Y., Liang, L.: Self-balancing federated learning with global imbalanced data in mobile systems. IEEE Transactions on Parallel and Distributed Systems 32(1), 59–71 (2021)
6. Fraboni, Y., Vidal, R., Kameni, L., Lorenzi, M.: Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 3407–3416. PMLR (18–24 Jul 2021), https://proceedings.mlr.press/v139/fraboni21a.html
7. Ghosh, A., Chung, J., Yin, D., Ramchandran, K.: An efficient framework for clustered federated learning. IEEE Transactions on Information Theory 68(12), 8076–8091 (2022)
8. He, C., Annavaram, M., Avestimehr, S.: Group knowledge transfer: Federated learning of large cnns at the edge. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 14068–14080. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper_files/paper/2020/file/a1d4c20b182ad7137ab3606f0e3fc8a4-Paper.pdf
9. Hu, Z., Shaloudegi, K., Zhang, G., Yu, Y.: Federated learning meets multi-objective optimization. IEEE Transactions on Network Science and Engineering 9(4), 2039–2051 (2022)

10. Huang, T., Lin, W., Wu, W., He, L., Li, K., Zomaya, A.Y.: An efficiency-boosting client selection scheme for federated learning with fairness guarantee. IEEE Transactions on Parallel and Distributed Systems 32(7), 1552–1564 (2021)
11. Jee Cho, Y., Wang, J., Joshi, G.: Towards understanding biased client selection in federated learning. In: Camps-Valls, G., Ruiz, F.J.R., Valera, I. (eds.) Proceedings of The 25th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 151, pp. 10351–10375. PMLR (28–30 Mar 2022), https://proceedings.mlr.press/v151/jee-cho22a.html
12. Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S.J., Stich, S.U., Suresh, A.T.: Scaffold: stochastic controlled averaging for federated learning. In: Proceedings of the 37th International Conference on Machine Learning. ICML'20, JMLR.org (2020)
13. Lai, F., Zhu, X., Madhyastha, H.V., Chowdhury, M.: Oort: Efficient federated learning via guided participant selection. In: 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21). pp. 19–35. USENIX Association (Jul 2021), https://www.usenix.org/conference/osdi21/presentation/lai
14. Li, C., Zeng, X., Zhang, M., Cao, Z.: Pyramidfl: a fine-grained client selection framework for efficient federated learning. In: Proceedings of the 28th Annual International Conference on Mobile Computing And Networking. p. 158–171. MobiCom '22, Association for Computing Machinery, New York, NY, USA (2022), https://doi.org/10.1145/3495243.3517017
15. Li, Q., He, B., Song, D.: Model-contrastive federated learning. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10708–10717 (2021)
16. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. In: Dhillon, I., Papailiopoulos, D., Sze, V. (eds.) Proceedings of Machine Learning and Systems. vol. 2, pp. 429–450 (2020), https://proceedings.mlsys.org/paper_files/paper/2020/file/1f5fe83998a09396ebe6477d9475ba0c-Paper.pdf
17. Liao, Y., Shen, X., Rao, H.: Analytic sensor rules for optimal distributed decision given k-out-of-l fusion rule under monte carlo approximation. IEEE Transactions on Automatic Control 65(12), 5488–5495 (2020)
18. Lyu, L., Yu, J., Nandakumar, K., Li, Y., Ma, X., Jin, J., Yu, H., Ng, K.S.: Towards fair and privacy-preserving federated deep models. IEEE Transactions on Parallel and Distributed Systems 31(11), 2524–2541 (2020)
19. Sattler, F., Müller, K.R., Samek, W.: Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. IEEE Transactions on Neural Networks and Learning Systems 32(8), 3710–3722 (2021)
20. Sattler, F., Wiedemann, S., Müller, K.R., Samek, W.: Robust and communication-efficient federated learning from non-i.i.d. data. IEEE Transactions on Neural Networks and Learning Systems 31(9), 3400–3413 (2020)
21. Tang, M., Ning, X., Wang, Y., Sun, J., Wang, Y., Li, H., Chen, Y.: Fedcor: Correlation-based active client selection strategy for heterogeneous federated learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10102–10111 (June 2022)
22. Thi Le, T.H., Tran, N.H., Tun, Y.K., Nguyen, M.N.H., Pandey, S.R., Han, Z., Hong, C.S.: An incentive mechanism for federated learning in wireless cellular networks: An auction approach. IEEE Transactions on Wireless Communications 20(8), 4874–4887 (2021)
23. Tu, L., Ouyang, X., Zhou, J., He, Y., Xing, G.: Feddl: Federated learning via dynamic layer sharing for human activity recognition. In: Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems. p. 15–28. SenSys '21, Association for Computing Machinery, New York, NY, USA (2021), https://doi.org/10.1145/3485730.3485946

24. Wang, H., Kaplan, Z., Niu, D., Li, B.: Optimizing federated learning on non-iid data with reinforcement learning. In: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications. pp. 1698–1707 (2020)
25. Wang, H., Qu, Z., Guo, S., Gao, X., Li, R., Ye, B.: Intermittent pulling with local compensation for communication-efficient distributed learning. IEEE Transactions on Emerging Topics in Computing 10(2), 779–791 (2022)
26. Wang, J., Chang, X., Mišić, J., Mišić, V.B., Wang, Y.: Pass: A parameter audit-based secure and fair federated learning scheme against free-rider attack. IEEE Internet of Things Journal 11(1), 1374–1384 (Jan 2024), http://dx.doi.org/10.1109/JIOT.2023.3288936
27. Wu, Q., Chen, X., Zhou, Z., Zhang, J.: Fedhome: Cloud-edge based personalized federated learning for in-home health monitoring. IEEE Transactions on Mobile Computing 21(8), 2818–2832 (2022)
28. Xu, J., Wang, H.: Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective. IEEE Transactions on Wireless Communications 20(2), 1188–1200 (2021)
29. Xue, Z., Wang, H.: Effective density-based clustering algorithms for incomplete data. Big Data Mining and Analytics 4(3), 183–194 (2021)
30. Yang, M., Wang, X., Zhu, H., Wang, H., Qian, H.: Federated learning with class imbalance reduction. In: 2021 29th European Signal Processing Conference (EUSIPCO). pp. 2174–2178 (2021)
31. Yao, X., Sun, L.: Continual local training for better initialization of federated models. In: 2020 IEEE International Conference on Image Processing (ICIP). pp. 1736–1740 (2020)
32. Yu, S., Chen, X., Zhou, Z., Gong, X., Wu, D.: When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5g ultradense network. IEEE Internet of Things Journal 8(4), 2238–2251 (2021)
33. Zhang, J., Cheng, X., Wang, C., Wang, Y., Shi, Z., Jin, J., Song, A., Zhao, W., Wen, L., Zhang, T.: Fedada: Fast-convergent adaptive federated learning in heterogeneous mobile edge computing environment. World Wide Web 25(5), 1971–1998 (Sep 2022), https://doi.org/10.1007/s11280-021-00989-x
34. Zhang, S., He, J., Liang, W., Li, K.: Mmds: A secure and verifiable multimedia data search scheme for cloud-assisted edge computing. Future Gener. Comput. Syst. 151(C), 32–44 (Feb 2024), https://doi.org/10.1016/j.future.2023.09.023

Shiwen Zhang received his Ph.D. degree from the College of Computer Science and Electronic Engineering, Hunan University, in 2016. He is currently an associate professor at the School of Computer Science and Engineering, Hunan University of Science and Technology. He is a member of IEEE and CCF. His research interests include identity authentication, security and privacy issues in Wireless Body Area Networks (WBAN), cloud computing, privacy protection, and information security.
Email: shiwenzhang@hnust.edu.cn.

Shuang Chen received a B.S. degree from Hunan University of Science and Technology (HNUST) in 2022 and am pursuing an M.S. degree from the School of Computer Science and Engineering at Hunan University of Science and Technology. Her research interests include clustering algorithms in federated learning, edge computing, and information security.

Wei Liang received a Ph.D. degree in computer science and technology from Hunan University in 2013. He was a Post-Doctoral Scholar at Lehigh University from 2014 to

2016. He is currently a Professor and the Dean of the School of Computer Science and Engineering at Hunan University of Science and Technology, China. He has authored or co-authored more than 140 journal/conference papers. His research interests include federated learning, edge computing, and blockchain.

Kuanching Li is a Professor at the School of Computer Science and Engineering, Hunan University of Science and Technology. Dr. Li has co-authored over 150 conference and journal papers, holds several patents, and serves as an associate and guest editor for various scientific journals. He has also held chair positions at several prestigious international conferences. His research interests include cloud and edge computing, big data, and blockchain technologies. Dr. Li is a Fellow of the Institution of Engineering and Technology (IET).

Arcangelo Castiglione is an associate professor at the Department of Computer Science, University of Salerno, Italy. He received a Ph.D. degree in Computer Science from the University of Salerno, Italy. He is an Associate Editor for several high-ranked journals, involved in several renowed international conference organizational roles, a member of the IEEE TC on Secure and Dependable Measurement, and a founding member of the IEEE TEMS TC on Blockchain and Distributed Ledger Technologies. His research mainly focuses on cryptography, network security, data protection, digital watermarking, and automotive security.

Junsong Yuan is a Professor and Director of the Visual Computing Lab at the Department of Computer Science and Engineering (CSE), State University of New York at Buffalo, USA. Before joining SUNY Buffalo, he was an Associate Professor (2015-2018) and a Nanyang Assistant Professor (2009-2015) at Nanyang Technological University (NTU), Singapore. He obtained his Ph.D. from Northwestern University in 2009, M.Eng. from the National University of Singapore in 2005, and B.Eng. from Huazhong University of Science Technology (HUST) in 2002. He received the Chancellor's Award for Excellence in Scholarship and Creative Activities from SUNY, Nanyang Assistant Professorship from NTU, and Outstanding EECS Ph.D. Thesis award from Northwestern University, and Best Paper Award from IEEE Trans. on Multimedia. He serves as Senior Area Editor of Journal of Visual Communication and Image Representation (JVCI), Associate Editor of IEEE Trans. on Pattern Analysis and Machine Intelligence (T-PAMI), IEEE Trans. on Image Processing (T-IP), IEEE Trans. on Circuits and Systems for Video Technology (T-CSVT), and Machine Vision and Applications (MVA). He also serves as General/Program Co-chair of ICME and Area Chair for CVPR, ICCV, ECCV, ACM MM, etc. He was elected as a faculty senator at both SUNY Buffalo and NTU. He is a Fellow of IEEE and IAPR.

Received: January 31, 2024; Accepted: April 24, 2025.