Multi-initial-Center Federated Learning with Data Distribution Similarity-Aware Constraint

Xiaoying Li[1,2], Xiaojun Chen[1(✉)], Shaopu Wang[1,2], Yangyang Ding[1], and Kaiyun Li[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{lixiaoying,chenxiaojun,wangshaopu,dingyangyang,likaiyun}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. Federated Learning (FL) has recently attracted high attention since it allows clients to collaboratively train a model while the training data remains local. However, due to the inherent heterogeneity of local data distributions, the trained model usually fails to perform well on each client. Clustered FL has emerged to tackle this issue by clustering clients with similar data distributions. However, these model-dependent clustering methods tend to perform poorly and be costly. In this work, we propose a distribution similarity-based clustered federated learning framework FedDSMIC, which clusters clients by detecting the client-level underlying data distribution based on the model's memory of training data. Furthermore, we extend the assumption about data distribution to a more realistic cluster structure. The center models are learned as good initial points to obtain common data properties in the cluster. Each client in a cluster gets a more personalized model by performing one step of gradient descent from the initial point. The empirical evaluation on real-world datasets shows that FedDSMIC outperforms popular state-of-the-art federated learning algorithms while keeping the lowest communication overhead.

Keywords: Clustered federated learning · Kullback-Leibler divergence · Model-Agnostic Meta-Learning · Non-IID data

1 Introduction

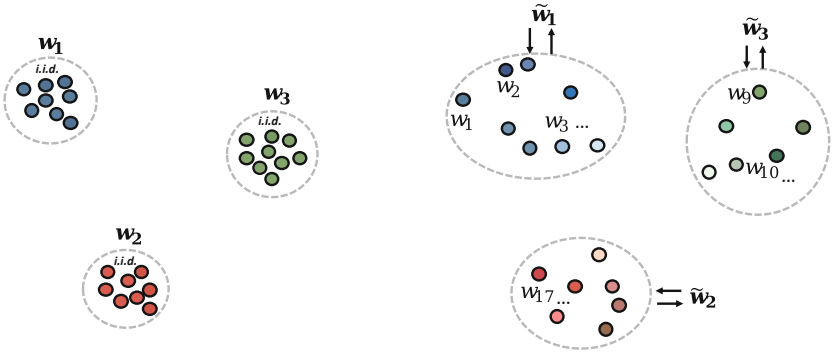
Federated learning (FL) is a promising distributed machine learning framework that can collaboratively train a joint model while keeping the data on the client side [19]. Classical FL trains a unique global model for all clients [20,22,27,33,34]. However, such global collaboration always fails to achieve good performance for individual clients since the data statistical heterogeneity, which is known as non-i.i.d. data [19,27,38]. In practice, clients usually have varying preferences. Consider the scenario for mobile device keyboards, certain emojis are used by one

W. Meng et al. (Eds.): ICA3PP 2022, LNCS 13777, pp. 752–772, 2023.
https://doi.org/10.1007/978-3-031-22677-9_41

demographic but not others. Therefore, it is necessary to provide personalized models for each client in FL.

A variety of personalized approaches have been proposed to tackle data heterogeneity, mainly from two perspectives: global model personalization and personalized models learning. The former first trains a global model and then fine-tune the global model locally [29,41]. However, the local distribution may be fairly different from the global distribution in highly personalized scenarios. Consequently, the relevant global model does not exist, and these approaches downgrade to each client learning only locally [13,32]. While the latter directly learns multiple individual personalized models. MOCHA [39] frames FL personalization as an MTL problem by exploiting penalization terms to capture relationships among clients. Unfortunately, it is usually tricky to simultaneously optimize multiple non-convex objectives determined by large neural networks.

To address the lack of the above studies, clustered FL groups clients into clusters and trains a model for each cluster, providing a trade-off between a purely global and personalized model. Several methods [3,38,42] clusters clients at the server-side based on the cosine similarity or l_2 distance of local model weights. Unfortunately, due to the high dimensionality and permutation invariance of neural network (NN) parameters, these methods often fail to cluster clients correctly. Other approaches [14,29] performs the cluster identities estimation at the client sides. In particular, k global models are randomly initialized, representing k clusters, and each client selects the model with the lowest loss on its local data. Clients that select the same model are assigned to a cluster. The methods improve the clustering performance, however, increasing the communication and computation burden for receiving and running multiple global models.



(a) Traditional clustered FL (b) Ours

Fig. 1. Comparison of traditional clustered FL and ours.

Moreover, the existing clustered FL methods are limited by the ideal assumption that each client belongs to a cluster with a specific data distribution. However, data heterogeneity is usually severe in the cross-device scenario with mas-

sive clients. Clients in a group can share one learning task, e.g., animals or vehicles classification [4], but the data distributions are still different (e.g., covariate shift, concept drift, and so forth [19]). We regard it as a more realistic cluster structure that clients in the same cluster are more loosely separated. Figure 1 depicts a comparison between traditional clustered FL and ours. Under the complicated data distribution, the client's data distribution information is not fully exploited using traditional clustered FL, which affects the clustering accuracy more severely.

In this paper, we present a novel clustered multi-task federated learning framework named FedDSMIC. Under the assumption of a realistic cluster structure, FedDSMIC reformulates the problem as an alternating minimization (AM) approach in the distributed setting, which optimizes the cluster assignment and minimizes the loss functions of the models alternatively. Specifically, FedDSMIC clusters clients based on Kullback-Leibler divergence between models' probability output distribution with respect to indicator samples on the server, which accurately detects the similarity of client-level underlying data distributions. Inspired by Model-Agnostic Meta-Learning (MAML), the goal of FedDSMIC is to learn the cluster model as a good initial point shared between all clients in the cluster, which performs well after each client updates it with respect to its loss function. The current or new clients in the cluster can quickly get their personalized models by performing one or a few steps of gradient descent from the initial point. This approach keeps all the benefits of the clustered FL architecture and leads to more personalized models for each client.

We summarise our main contributions as:

- We propose a dynamic clustered federated learning framework, which clusters clients by detecting the client-level underlying distribution based on the model's parameter memory for the training data, improving the clustering accuracy in high data heterogeneity.
- We illustrate the limitation of sharing one model in the cluster and introduce a two-step learning method, which builds an initial center model to capture the intra-cluster common information and learns personalized models for each client to acquire unique features, improving the personalized model performance.
- Extensive experiments conducted on five real-world datasets demonstrate FedDSMIC outperforms other state-of-the-art methods with fewer communication rounds and computational consumption.

2 Related Work

Here, we mainly review the existing works from two aspects: classic federated learning and personalized federated learning.

2.1 Classic Federated Learning

The classic federated learning [33,40] trains a single global model to minimize an empirical risk function over the union of the data across all clients. However,

various studies have shown that non-i.i.d. decentralized data leads to statistical challenges such as model weights divergence [44], data distribution biases [16], and a drifted global model that is slow to converge even unguaranteed convergence [27]. Li *et al.* [27] proposed FedProx, which adds a proximal term to the local objective function to reduce the gap between local and global models. SCAFFOLD [20] introduces control variates to correct the client drift in its local updates. FedGen [46] sets a generator on the server to ensemble client information and regulate local training using the learned knowledge. While the above work focus on building a robust global model across non-i.i.d. data, they do not directly address local model performance relevant to individual clients.

2.2 Personalized Federated Learning

Global Model Personalization. A natural approach for personalized FL is learning a global model and fine-tuning parameters on each client's local dataset [1,9,29,30,41]. The global model serves as a starting point for a few-shot adaptation for each client. Therefore, a class of algorithms referred to as meta-learning has been developed to train a more suitable global model for local customization [6,11,18,21]. Interpolation of global and local model [15,24,29] build personalized models for clients by combining the global model and the local model. A good global model is critical, as the personalization performance directly depends on the generalization performance of the global model. Unfortunately, it is difficult to obtain a good global model in high data heterogeneity.

Learning Personalized Models. Multi-Task federated learning methods treat clients as different tasks and train personalized models for each client. Simth *et al.* [39] proposed MOCHA that adds a penalization term to capture relationships between clients. However, it only learns simple models because of the complex penalization term. Other MTL-based approaches [17,26] are able to train more general models at the cost of considering simpler penalization terms. Therefore, it is tricky to optimize the complex objective function, and all of them lack statistical assumptions about local data distributions.

Clustered FL assumes that the clients can be partitioned into k clusters, representing k different distributions. CFL [38] recursively separate clients with incongruent optimization directions by the cosine similarity of the parameter updates. FedSEM [42] uses a l_2 distance-based stochastic expectation maximization (EM) algorithm, which ignores l_2 distance often suffer in high-dimension, low-sample-size (HDLSS) situation [37]. Briggs *et al.* [3] propose an agglomerative hierarchical clustering method named FL+HC, which relies on iterative calculating the pairwise distance between all clusters. The above parameter-based similarity measures always fail to cluster clients correctly because of the permutation invariance of NN parameters, i.e., for any given NN, many variants of it only differ in the ordering of parameters. To overcome the drawback, IFCA [14,29] divides the clients into clusters with a center model that can minimize their loss values while requiring each client to train all k global models per

round. Therefore, the computational and communication efficiency will become bottlenecks when IFCA is applied to a large-scale FL system.

Our clustering method is similar to IFCA but allows clustering on the server for less communication. We focus on detecting the client's underlying data distribution without explicitly using model parameters. Finally, unlike previous work, we learn a personalization model for each client based on the center model to cope with more complicated heterogeneous data.

3 Preliminaries

Consider M clients, each client i has a private labeled dataset $\mathcal{Z} = \{z_i^{(n)} = (\boldsymbol{x}_i^{(n)}, y_i^{(n)})_{n=1}^{n_i}\}$, where n_i is the training dataset size of client i. It is generated according to the local distribution $\mathcal{D}_i = \{\mathcal{X}_i\ , \mathcal{Y}_i\}$, where $\mathcal{X}_i$ and $\mathcal{Y}_i$ denote the input features and corresponding labels, respectively. Each private dataset $\mathcal{Z}_i$ will be used to train a local supervised learning model $\boldsymbol{w}_i : \mathcal{X}_i \to \mathcal{Y}_i$. We define $f_i : \mathbb{R}^d \to \mathbb{R}$ as the expected loss over the data distribution with respect to client i, i.e.,

$$f_i(\boldsymbol{w}) := \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_i}[l_i(\boldsymbol{w}_i; \boldsymbol{x}, y)] \tag{1}$$

where $\boldsymbol{w}_i \in \mathbb{R}^d$ is the parameter space, $l_i(\boldsymbol{w}_i; \boldsymbol{x}, y)$ is the error of model $\boldsymbol{w}_i$ in predicting the true label $y \in \mathcal{Y}_i$ given the input $\boldsymbol{x} \in \mathcal{X}_i$. The goal of vanilla FL (FedAvg) is to solve the objective function

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} F(\boldsymbol{w}) := \sum_{i=1}^{M} \frac{n_i}{n} f_i(\boldsymbol{w}). \tag{2}$$

where $n = \sum_{i=1}^{M} n_i$ is the total training dataset size. FedAvg optimizes (2) by the local updates of clients and the aggregation of the server alternately. At each communication round t, the server broadcasts the latest global model $\boldsymbol{w}^t$ to all clients and selects a random subset M_t of M clients to participate in this round. Client i optimizes the loss function based on the local data by its local solver (SGD) with several iterations or epochs and gets the updated local model $\boldsymbol{w}_i^{t+1}$. Then, the server takes a weighted average of all local resulting model parameters $\{\boldsymbol{w}_1^{t+1}, \boldsymbol{w}_2^{t+1}, ..., \boldsymbol{w}_{M_t}^{t+1}\}$ into a global one $\boldsymbol{w}^{t+1}$ and finish the current round. However, the different distributions of the local data $\mathcal{D}_i$ lead to different local model parameters, failing to converge to a stable optimal global solution $\boldsymbol{w}$.

The existing clustered FL framework usually assumes that clients can be clustered into several groups to address the data heterogeneity. Besides, clients in the cluster share the same data distribution and optimization goal.

4 Framework

4.1 Problem Formulation

We assume a non-i.i.d. data distribution with a clustering structure: the data distribution of clients in a cluster is similar but still different, which we discussed in the Introduction. Under this assumption, we aim first to build several

clusters and learn the initial center models to capture the intra-cluster common information, then learn personalized models for each client to discover knowledge different from others in the cluster. Specifically, we formulate a clustered multi-task federated learning problem as follows:

$$\min_{\{\boldsymbol{w}_i\},\{\widetilde{\boldsymbol{w}}_c\},\{\boldsymbol{u}_i^c\}} \sum_{c=1}^{C} L_c(\widetilde{\boldsymbol{w}}_c) - \sum_{c=1}^{C}\sum_{i=1}^{M} \boldsymbol{u}_i^c S(\boldsymbol{w}_i, \widetilde{\boldsymbol{w}}_c) \tag{3}$$

$$L_c(\widetilde{\boldsymbol{w}}_{\boldsymbol{c}}) = \sum_{i=1}^{M_c}\sum_{n=1}^{n_i} l_i(\boldsymbol{w}_i; \mathbf{x}_i^{(n)}, y_i^{(n)}), i \in G_c \tag{4}$$

where M is the number of total clients, M_c is the number of clients in cluster G_c, C is the number of clusters. $\widetilde{\boldsymbol{w}}_c$ is the parameters of the center model for cluster G_c. In addition, $S(\boldsymbol{w}_i, \widetilde{\boldsymbol{w}}_c)$ denotes the similarity of local model $\boldsymbol{w}_i$ and the center model $\widetilde{\boldsymbol{w}}_c$. $\{\boldsymbol{u}_i^c\}$ denotes the cluster assignment, with $\boldsymbol{u}_i^c = 1$ if the clients C_i belongs to cluster G_c and $\boldsymbol{u}_i^c = 0$ otherwise.

In the above formulation, the first term (4) is the sum of loss functions of all center models, and the loss of each center model is given by the sum of empirical errors across all clients in this cluster. The second term is the sum of similarities between local models and center models, which can be seen as K-means clustering to maximize the intra-cluster similarity. Here we have three variables $\{\boldsymbol{w}_i\}, \{\widetilde{\boldsymbol{w}}_c\}, \{\boldsymbol{u}_i^c\}$ to be solved under federated settings. Alternating minimization [2] is the general approach to solving such a non-convex optimization problem. Specifically, we minimize the loss functions based on $\boldsymbol{w}_i$ and $\widetilde{\boldsymbol{w}}_c$, and estimate the cluster assignment $\boldsymbol{u}_i^c$ by alternatively fixing one and optimizing another until convergence.

We elaborate our alternating optimization with a formal shown in Algorithm 1. As shown in Fig. 2, FedDSMIC is a dynamic clustered federated learning with four main processes. FedDSMIC starts with C randomly initialized center models and M local models. At each communication round t, each participating client i updates locally and sends the local model $\boldsymbol{w}_i^{t+1}$ to the server. The server computes the similarity of clients based on Kullback-Leibler divergence between the predictions of local models $\{\boldsymbol{w}_1^{t+1}, \boldsymbol{w}_2^{t+1}, ..., \boldsymbol{w}_{M_t}^{t+1}\}$ and center models $\{\widetilde{\boldsymbol{w}}_1^t, ..., \widetilde{\boldsymbol{w}}_c^t\}$ on "indicator samples". Then the server updates $\boldsymbol{u}_i^c$ and clusters clients to maximize the intra-cluster similarity of the models. Finally, each center model $\widetilde{\boldsymbol{w}}_c^t$ is updated by the weighted average of local models in this cluster to $\widetilde{\boldsymbol{w}}_c^{t+1}$ and sent to the intra-cluster clients. After training T rounds, the personalized models of clients are obtained by performing one or a few gradient descents from their corresponding center model. FedDSMIC also allows new clients to get their personalized models easily. In Sect. 4.2 and Sect. 4.3, we present the optimization of cluster structure and the optimization of the models, respectively.

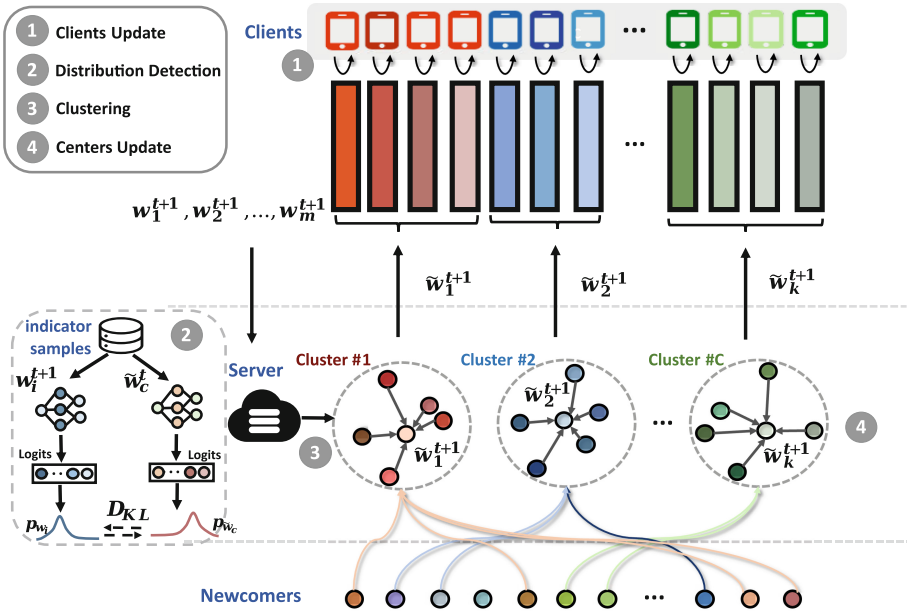


Fig. 2. An overview of FedDSMIC

4.2 Cluster Structure Optimization

Based on the guideline of FedDSMIC, we aim to first cluster the clients with similar data distribution into one group without access to the local data. When $\boldsymbol{w}_i$ and $\widetilde{\boldsymbol{w}}_{\boldsymbol{c}}$ is fixed, problem (3) w.r.t $\boldsymbol{u}_{\boldsymbol{i}}^{\boldsymbol{c}}$ can be seen as K-means clustering on the local's models. The crucial challenge is how to quantify the similarity of models for accurately clustering clients with similar underlying data distributions.

Similarity of Models. To tackle the challenge, we leverage the memory (learned parameters) of the model and get the similarity of models from the underlying data distribution on which it is trained. Knowledge distillation performs knowledge transfer by reducing the distance between the predicted output probability distribution of the student model and the teacher model on the same dataset [43]. Inspired by this, we set up some "indicator samples" on the server and then feed the samples into the model to get predictions to reflect the training data distribution of the model. The assumption for the distribution of "indicator samples" is independent and identically distributed. In the experiment, the samples are randomly sampled from the raw dataset (e.g., 10 samples per class) before partitioning the client data. In practice, these samples can be sampled from the relevant public datasets or a small sharing of datasets from clients under the premise of privacy protection. In addition, we will demonstrate the effect of indicator sample size on describing clients in Sect. 5.

Algorithm 1. FedDSMIC

Input: numbers of clusters C, initialize $\{\widetilde{\boldsymbol{w}}_c\}_{c=1}^{C}$, $\{\boldsymbol{w}_i\}_{i=1}^{M}$, number of communication rounds T
Output: $\{\boldsymbol{w}_i\}_{i=1}^{M}, \{\widetilde{\boldsymbol{w}}_c\}_{c=1}^{C}$
for $t = 0, ..., T$ **do**
  $M_t \leftarrow$ random subset of M clients
  Clients:
  for each client $i \in M_t$ **in parallel do**
    $\boldsymbol{w}_i^{t+1} \leftarrow$ **ClientUpdate**$(i, \boldsymbol{w}_i^t)$
  end for
  Server:
  for each cluster $c = 1, ..., C$ **do**
    Calculate $Dis(\boldsymbol{w}_i^{t+1}, \widetilde{\boldsymbol{w}}_c^t)$ as in (6)
    Update $\boldsymbol{u}_i^c$ using $Dis(\boldsymbol{w}_i^{t+1}, \widetilde{\boldsymbol{w}}_c^t)$ as in (7)
  end for
  Group devices into G_c using $\boldsymbol{u}_i^c$
  Update $\widetilde{\boldsymbol{w}}_c^{t+1}$ by $\widetilde{\boldsymbol{w}}_c^{t+1} = \sum_{i=1}^{M_c} \frac{n_i}{n_c} \boldsymbol{w}_i^{t+1}, i \in G_c$
  for each cluster $c = 1, ..., C$ **do**
    for $i \in$ cluster G_c **do**
      Send $\widetilde{\boldsymbol{w}}_c^{t+1}$ to client i
    end for
  end for
end for

Algorithm 2. ClientUpdate

Input: the corresponding cluster center model from server $\widetilde{\boldsymbol{w}}_c^t \rightarrow \boldsymbol{w}_i^{t,0}$
Output: $\boldsymbol{w}_i^{t+1}$
$\mathcal{B} \leftarrow$ split D_i into batches of size $\mathcal{B}$
for k: 1 to τ **do**
  Set $\widehat{\boldsymbol{w}}_i^{t+1,k+1} = \boldsymbol{w}_i^{t+1,k} - \eta \widetilde{\nabla} l_i \left(\boldsymbol{w}_i^{t+1,k}, D_i^k \right)$
  Set $\boldsymbol{w}_i^{t+1,k+1} = \boldsymbol{w}_i^{t+1,k} - \varepsilon \widetilde{\nabla} l_i \left(\widehat{\boldsymbol{w}}_i^{t+1,k+1}, D_i^{'k} \right)$
end for
$\boldsymbol{w}_i^{t+1} = \boldsymbol{w}_i^{t+1,\tau}$

Let p^k be the probability belonging to each class k for an input sample x given by a neural network w, which is computed as:

$$p_w^k(x) = \frac{\exp\left(z_1^k\right)}{\sum_{k=1}^{K} \exp\left(z_1^k\right)} \tag{5}$$

where the logit z^k is the output of the pre-softmax layer of model w on the data x. The final layer of a recognition model is a fully connected layer with a softmax non-linearity. Each neuron in this layer corresponds to a class k, and its activation value is treated as the probability that the model predicts for that class. The weights connecting the previous layer to this neuron w^k can be considered the template of the label k learned by the network [35]. The

predicted label probability is proportional to the alignment of the pre-final layer's output with the template w^k. In other words, the value of trained weights w^k increases with the marginal density of label-k training data $p(y)^k$. Furthermore, the trained model has a higher probability of predicting the label-k sample as class k.

Therefore, we leverage the output probability distribution of the model to approximate the data distribution on which it was trained. To quantify the distance of the local model's and center model's predictions, we use the Kullback Leibler (KL) Divergence. The KL distance from p_{w_i} of the local model $\boldsymbol{w}_i$ to $p_{\widetilde{w}_c}$ of the center model $\widetilde{\boldsymbol{w}}_{\boldsymbol{c}}$ is computed as:

$$\begin{aligned} & D_{\mathrm{KL}}[\sigma(f(\boldsymbol{w}_i, \boldsymbol{x}_o)), \sigma(f(\widetilde{\boldsymbol{w}}_c, \boldsymbol{x}_o))] \\ = & \sum_{n_o=1}^{N_o} \sigma(f(\boldsymbol{w}_i, \boldsymbol{x}_o)) log \frac{\sigma(f(\boldsymbol{w}_i, \boldsymbol{x}_o))}{\sigma(f(\widetilde{\boldsymbol{w}}_c, \boldsymbol{x}_o))} \\ = & \sum_{n_o=1}^{N_o} \sum_{k=1}^{K} p_{w_i}^k(\boldsymbol{x}_o) \, log \frac{p_{w_i}^k(\boldsymbol{x}_o)}{p_{\widetilde{w}_c}^k(\boldsymbol{x}_o)} \end{aligned} \tag{6}$$

where the logits $f(\boldsymbol{w}, \boldsymbol{x}_o)$ is the output of the pre-softmax layer of model $\boldsymbol{w}$ on the indicator samples $\boldsymbol{x}_o$. σ is the non-linear activation (usually the softmax function for multi-class classification) applied to such logits. N_o is the number of indicator samples, and K is the number of classes (labels).

Cluster Identity Estimation. The server computes the KL divergence between local models and center models by (6), and obtain a KL divergence matrix $Dis \in \mathbb{R}^{M \times C}$, with $Dis_{i,j} = D_{\mathrm{KL}}[\sigma(f(\boldsymbol{w}_i, \boldsymbol{x}_o)), \sigma(f(\widetilde{\boldsymbol{w}}_j, \boldsymbol{x}_o))]$. As the smaller $Dis_{i,j}$ corresponds to the greater similarity $S(\boldsymbol{w}_i, \widetilde{\boldsymbol{w}}_j)$ in (3), the server updates the cluster assignment vector $\boldsymbol{u}_i^c$ by (7).

$$\boldsymbol{u}_i^c = \begin{cases} 1, & \text{if } c = \operatorname{argmin}_j Dis_{i,j} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

The efficiency analysis of our clustering method is as follows. Suppose the time of model inference is t, the total number of parameters of the model is d.

- **Computation.** In FedDSMIC, the time to get the probability distributions is $(M+C)*t$. In IFCA, the time to estimate the cluster identities is $M*C*t$ (each client trains C global models locally). The time complexity of FedDSMIC constantly grows with the number of clients M, while IFCA grows linearly. Therefore, our method has lower computational complexity, especially when M is large.
- **Communication.** In FedDSMIC, the server sends the cluster model to the corresponding clients in this cluster and receives M local models, so the traffic per round is $(M+M)*d$. In IFCA, the server sends C cluster models to each client, so the traffic per round is $(C*M+M)*d$, which is $(1+C)/2$ times more than ours. As the modern neural network model grows, the communication consumption of IFCA will increase.

4.3 Model Weights Optimization

Now $\{\boldsymbol{u}_i^c\}$ is fixed, the optimization of local models $\{\boldsymbol{w}_i\}$ and center models $\{\widetilde{\boldsymbol{w}}_c\}$ are as follows. The goal of FedDSMIC is to find a good initial cluster center model for representing the common information in this cluster. The cluster model performs well on all the clients in the cluster after each client updates it with respect to local data. We use one step of gradient descent from the initial model for computational efficiency. Therefore, the first term (4) in (3) can be rewritten as

$$L_c\left(\boldsymbol{w}\right) = \sum_{i=1}^{M_c} \sum_{n=1}^{n_i} l_i\left(\boldsymbol{w} - \alpha \nabla l_i\left(\boldsymbol{w}\right); \mathbf{x}_i^{(n)}, y_i^{(n)}\right) \tag{8}$$

for all $c \in C$. The optimal solution of $\boldsymbol{w}$ of $L_c\left(\boldsymbol{w}\right)$ is the learned cluster center model $\widetilde{\boldsymbol{w}}_c$. Equation (8) maintains the advantages of clustered FL, and further captures the difference between the clients in the cluster. To solve (8), we redefined the local function of client C_i as

$$g_i(\boldsymbol{w}) := l_i\left(\boldsymbol{w} - \eta \nabla l_i(\boldsymbol{w})\right) \tag{9}$$

where η is the learning rate. The gradient of $g_i(\boldsymbol{w}_i)$ is computed as

$$\nabla g_i(\boldsymbol{w}) = \left(I - \eta \nabla^2 l_i(\boldsymbol{w})\right) \nabla l_i\left(\boldsymbol{w} - \eta \nabla l_i(\boldsymbol{w})\right) \tag{10}$$

For computationally efficient, we use the first-order approximations of (10), i.e., the second-order gradient $\nabla^2 l_i(\boldsymbol{w})$ is approximated to zero. Then Eq. (10) can be rewritten as

$$\nabla g_i(\boldsymbol{w}) = \nabla l_i\left(\boldsymbol{w} - \eta \nabla l_i(\boldsymbol{w})\right) \tag{11}$$

Similar to Stochastic Gradient Descent (SGD), we take a batch of data $\mathcal{D}_i$ to obtain the unbiased estimate $\tilde{\nabla} l_i(\boldsymbol{w})$ of $\nabla l_i(\boldsymbol{w})$

$$\tilde{\nabla} l_i\left(\boldsymbol{w}\right) := \frac{1}{|\mathcal{D}_i|} \sum_{(\boldsymbol{x}, y) \in \mathcal{D}_i} \nabla l_i(\boldsymbol{w}; \boldsymbol{x}, y). \tag{12}$$

At each communication round t, the server sends the cluster center model $\widetilde{\boldsymbol{w}}_c^t$ to the corresponding clients in this cluster. Each client C_i in the cluster sets the initial parameters of the local model $\boldsymbol{w}_i^{t+1,0} = \widetilde{\boldsymbol{w}}_c^t$. Then client C_i performs τ steps of stochastic gradient descent locally with respect to g_i. The number of local iterations is τ, then the local updates sequence $\left\{\boldsymbol{w}_i^{t+1,k}\right\}_{k=0}^{\tau}$ are updated by

$$\boldsymbol{w}_i^{t+1,k+1} = \boldsymbol{w}_i^{t+1,k} - \varepsilon \tilde{\nabla} g_i\left(\boldsymbol{w}_i^{t+1,k}\right) \tag{13}$$

where ε is the local learning rate, $\tilde{\nabla} g_i\left(\boldsymbol{w}_i^{t+1,k}\right)$ is an estimate of $\nabla g_i\left(\boldsymbol{w}_i^{t+1,k}\right)$ in (11). The stochastic gradient $\tilde{\nabla} g_i\left(\boldsymbol{w}_i^{t+1,k}\right)$ for all local iterates is computed using independent batches $D_i^k, D_i^{'k}$ as follows

$$\tilde{\nabla} g_i\left(\boldsymbol{w}_i^{t+1,k}\right) = \tilde{\nabla} l_i\left(\boldsymbol{w}_i^{t+1,k} - \eta \bar{\nabla} l_i\left(\boldsymbol{w}_i^{t+1,k}, D_i^k\right), D_i^{'k}\right). \tag{14}$$

Actually, the updates in (13) can be implemented in two stages: first, we compute the first-order update value $\widehat{\boldsymbol{w}}_i^{t+1,k+1}$ as in (15), and then compute the final updated value $\boldsymbol{w}_i^{t+1,k+1}$ as in (16)

$$\widehat{\boldsymbol{w}}_i^{t+1,k+1} = \boldsymbol{w}_i^{t+1,k} - \eta \widetilde{\nabla} l_i \left(\boldsymbol{w}_i^{t+1,k}, D_i^k \right) \tag{15}$$

$$\boldsymbol{w}_i^{t+1,k+1} = \boldsymbol{w}_i^{t+1,k} - \varepsilon \widetilde{\nabla} l_i \left(\widehat{\boldsymbol{w}}_i^{t+1,k+1}, D_i^{'k} \right) \tag{16}$$

The steps of client local update are depicted in Algorithm 2. Our solution procedure follows the previous work [10–12] with convergence guarantees. After the local models are updated, each cluster model is obtained by the weighted average of the client models in this cluster as:

$$\widetilde{\boldsymbol{w}}_c^{t+1} = \sum_{i=1}^{M_c} \frac{n_i}{n_c} \boldsymbol{w}_i^{t+1}, i \in G_c \tag{17}$$

where n_c is the total data size of cluster G_c. Benefit from the properties of meta-learning [12], FedDSMIC allows an unseen client, i.e., a client $C_{new} \notin [M]$ arriving after the distributed training procedure, to easily learn its personalized model. Each new client simply first train on its local dataset for a few epochs and choose the most similar cluster center model through (6) and (7). Then the client obtains its personalized model by performing one or few steps with respect to its local data.

5 Experiments

5.1 Datasets and Baselines

Datasets and Models. We evaluate our algorithm on five federated benchmark datasets: handwritten digits recognition (MNIST [25]), handwritten characters recognition (EMNIST [7] and FEMNIST [5]), and image classification (CIFAR10 and CIFAR100 [23]). We train a convex multinomial logistic regression (MCLR) model on MNIST, a CNN in LEAF [5] on EMNIST and FEMNIST, and MobileNet-v2 [36] on CIFAR10 and CIFAR100. Table 1 summarizes datasets, models, and the number of clients. For all datasets, we randomly split each local dataset into training (80%) and test (20%) sets.

Table 1. Datasets and models setting

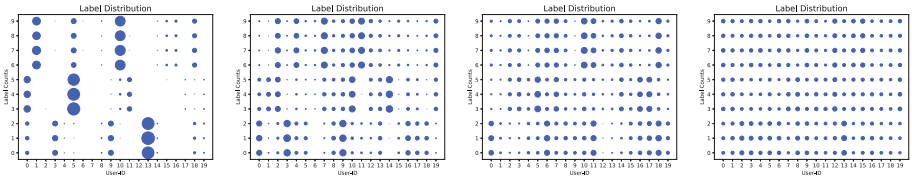
Dataset	Clients	Total samples	Model
MNIST [25]	100	70000	MCLR
EMNIST [7]	100	81415	2-layer CNN + 2-layer FFN
FEMNIST [5]	287	61049	2-layer CNN + 2-layer FFN
CIFAR10 [23]	80	30000	MobileNet-v2 [36]
CIFAR100 [23]	125	30000	MobileNet-v2 [36]

Baselines

- Local: To make the experiment more comprehensive, we report the performance of a naive personalized method named Local that trains only on the local dataset without collaboration.
- FedAvg [33]: the vanilla federated learning framework.
- FedProx [27]: a popular federated learning optimizer which adds a quadratic penalty term to the local objective.
- IFCA [14]: a hypothesis-based CFL framework that client selects the model with minimal empirical loss.
- FedSEM [42]: an l_2 distance-based CFL framework that minimizes the expectation of discrepancies between local models and center models stochastically.
- PerFedAvg [11]: a personalized method that finds one initial shared model for all clients.
- FedDS: clustering-only of our method that the clients in a cluster share the same model.

5.2 Experimental Setting

Client Heteroeneity. For FEMNIST, the raw data is naturally non-i.i.d distributed since writers who write the same words have different stroke widths. For MNIST, EMNIST, CIFAR10 and CIFAR100, similar to prior arts [17,31], we simulate a data distribution with a clustered structure that satisfies our assumption. First we divide all classes (labels) to C clusters, and then simulate a heterogeneous partition into M clients by sampling $p_c \sim Dir_I(\alpha)$ and allocating a $p_{c,i}$ proportion of the training instances of cluster c to local client i, in which a smaller α indicates higher data heterogeneity. The clients in the same cluster have relatively similar data distributions but are different from each other. We visualize the effects of adopting different α on the statistical heterogeneity for the MNIST dataset with M = 20 in Fig. 3.



(a) $\alpha = 0.05$ (b) $\alpha = 0.5$ (c) $\alpha = 1.0$ (d) $\alpha = 10.0$

Fig. 3. Visualization of a realistic cluster structure simulation among clients on MNIST dataset, where the x-axis indicates client IDs, the y-axis indicates class labels, and the size of scattered points indicates the number of training samples for a label available to that client.

Implementation. Unless otherwise mentioned, the number of clusters is 3 for all cluster-based methods, and all clients participate in each round, training occurred over 300 communication rounds for MNIST and FEMNIST, and 200 rounds for all other datasets. We use the learning rate 0.01, the mini-batch size $B = 32$, the local updating steps $\tau = 20$ and SGD with momentum $= 0.9$ for all algorithms. In addition, we extract a piece of data from each class of the dataset as indicator samples before assigning the data to the client.

Evaluation Metrics. Like previous research on personalized federated learning [11,39], we evaluate the performance of each personalized model on the local test dataset. In FedAvg and FedProx, we evaluate the global model based on the test set of all clients. In IFCA and FedSEM, we evaluate the cluster center model based on the test set of the clients in this cluster. Given the multiple accuracies of clients with different data sizes, we report the average weighted accuracy with weights proportional to local dataset sizes. Note that the heterogeneity will affect the convergence, resulting in more significant fluctuations in model accuracy during the training process. Therefore, we report the average of the top-5 test accuracy rates.

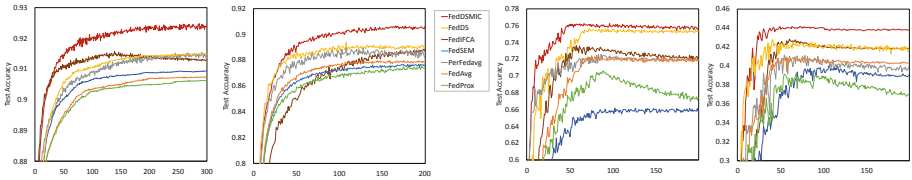
5.3 Effectiveness of Proposed Framework

Average Performance of Personalized Models. The comparison results with respect to the average performance of personalized models are shown in Table 2 and Fig. 4. We have the following findings from the results:

- Overall, FedDSMIC obtains the best performance across all datasets and has the most rapid learning curves. Notably, FedDSMIC improves test accuracy by around 3% to 5% on handwritten digit recognition tasks while improving accuracy more than by +10% on CIFAR10 and CIFAR100, which are more complex classification tasks. The results demonstrate the effectiveness of FedDSMIC, especially when the data distribution is highly complicated.
- As one of the most competitive baselines, IFCA can achieve similar performance to FedDS in specific settings, but the convergence is slower than FedDS. This result implies that our method fully leverages information about underlying data distribution, whose effect is more explicit.
- FedSEM performs worst in all settings compared with FedDSMIC and IFCA, which can be interpreted as failing to cluster clients correctly. FeSEM always clusters all clients into a group. Therefore, it has been downgraded to one optimization direction and behaves similarly to FedAvg.

Table 2. Average test accuracy across clients/bottom test accuracy under different settings. For MNIST and EMNIST, a smaller α indicates higher heterogeneity. For FEMNIST, r denotes the ratio between active users and total users. For CIFAR10 and CIFAR100, τ denotes the local training steps, E denotes the local training epochs.

Dataset	Setting	Local	FedAvg	FedProx	IFCA	FedSEM	PerFedAvg	FedDS	FedDSMIC
MNIST	$\alpha = 0.05$	94.0/65.5	89.7/76.7	89.6/76.7	92.8/75.3	90.6/72.1	92.0/77.4	92.5/75.0	**94.2/79.0**
	$\alpha = 0.5$	88.9/73.5	90.0/83.4	90.7/83.4	90.7/84.8	90.1/82.2	91.5/85.0	91.5/82.9	**92.5/86.0**
	$\alpha = 1.0$	87.5/68.3	90.4/84.2	90.3/84.0	90.7/83.1	90.4/84.1	91.8/**86.6**	90.7/84.2	**92.0**/85.4
EMNIST	$\alpha = 0.05$	90.1/48.4	86.6/76.5	85.7/81.3	92.3/72.7	84.6/77.0	92.0/84.1	92.7/82.1	**93.2/84.4**
	$\alpha = 0.5$	78.6/31.2	88.0/80.9	87.6/79.3	89.1/80.7	87.7/81.9	88.9/81.9	89.4/80.2	**90.7/83.0**
	$\alpha = 1.0$	75.2/55.9	87.6/82.5	87.0/81.7	87.1/80.8	87.5/82.5	88.2/**84.6**	88.9/83.9	**89.0**/84.1
FEMNIST	r = 0.2	69.2/50.8	81.7/64.2	81.5/63.1	**87.7/67.2**	79.8/63.8	83.8/72.3	87.2/65.7	87.4/67.1
	r = 0.5	69.4/47.7	84.0/53.8	82.5/48.1	87.1/67.5	83.4/59.4	86.8/67.4	87.0/62.3	**87.1/68.0**
	r = 1.0	69.5/48.6	83.9/51.0	82.3/45.1	87.0/67.3	84.1/56.0	86.8/67.9	87.0/62.5	**87.4/68.2**
CIFAR10	$\tau = 10$	36.8/18.0	70.5/44.3	69.1/47.0	73.1/49.7	64.4/38.9	73.4/53.4	75.3/**56.4**	**75.5**/55.6
	$\tau = 20$	35.7/17.9	72.3/47.5	70.5/45.3	73.5/44.8	66.2/42.4	72.5/52.6	75.6/52.1	**76.2/60.5**
CIFAR100	E = 3	13.8/3.2	40.9/10.0	39.1/10.9	42.7/9.6	39.9/7.0	41.1/6.2	42.7/7.8	**44.9/11.2**
	E = 6	13.1/4.5	41.8/10.8	36.6/10.5	43.9/10.5	40.5/8.9	40.9/3.4	43.7/9.2	**44.8/11.6**



(a) MNIST (b) EMNIST (c) CIFAR10 (d) CIFAR100

Fig. 4. Average test accuracy vs. communication rounds on MNIST, EMNIST, CIFAR10, CIFAR100, $\alpha = 0.5$ for all datasets, $C = 3$ for FedDSMIC, FedDS, IFCA and FeSEM.

Fairness Across Clients. This work discusses heterogeneous data scenarios in horizontal federated learning, so we mainly focus on performance fairness, i.e., accuracy is evenly distributed among clients [45]. Unfair performance could result from learning excellent models for some clients at the expense of poor models for other clients [31]. Consequently, we show the average of the five worst accuracies called bottom accuracy in Table 2 (the minimum accuracy is particularly noisy when some local datasets are too small). FedDSMIC ensures that clients with the worst personalized model are better than other methods.

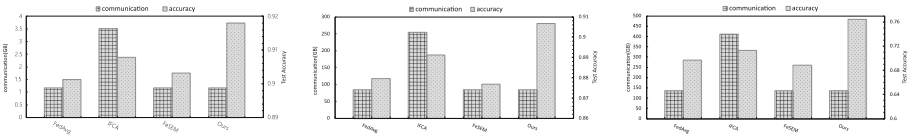
Generalization to Unseen Clients. Table 3 shows that FedDSMIC allows new clients to learn a personalized model and consistently outperforms other CFL-based methods. As discussed in Sect. 4.3, FedDSMIC allows new clients arriving after the distributed training to learn their personalized models quickly. To evaluate the quality of unseen clients' personalized models, we performed an experiment where only 80% of the clients participate to the training. The

remaining 20% join the system after training and get their personalized model by our method.

Table 3. Average test accuracy across clients unseen at training

Dataset	FedAvg	IFCA	FedSEM	FedDMIC
MNIST	90.92	90.62	88.93	**91.52**
EMNIST	81.89	81.71	82.56	**86.18**
FEMNIST	73.84	74.76	72.94	**75.75**
CIFAR10	35.59	36.89	42.45	**57.74**
CIFAR100	13.8	14.37	10.26	**15.43**

Communication Consumption. As shown in Fig. 5, FedDSMIC has the lowest communication consumption and highest accuracy on different network structures. FedDSMIC and FedSEM perform cluster estimation on the server side without sending additional models, therefore having the same traffic as FedAvg. As discussed in Sect. 4.2, IFCA requires the server to send all center models to each client, thus consuming the most communication.

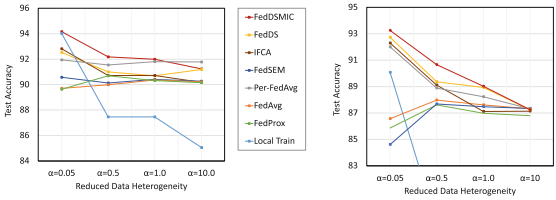


(a) MLP-MNIST (b) CNN-EMNIST (c) Mobilenet-v2 -CIFAR10

Fig. 5. Communication consumption of FL frameworks on different network structures for 300 rounds.

5.4 Sensitivity Analysis

Impacts of Data Heterogeneity. As shown in Fig. 6, FedDSMIC performs better than others under various data heterogeneity settings, and the gain of FedDSMIC is more notable when the data distributions are highly heterogeneous (with a small α). This result verifies our motivations since the advantage of FedDSMIC is building personalized models for each client based on its corresponding cluster model, which mitigates the data heterogeneity. This advantage is otherwise not obtained by other baselines. For MNIST, Per-FedAvg has the best performance at the i.i.d setting ($\alpha = 10$) because it may utilize information from more training data. For the more complex dataset EMNIST, the performance gain of our approach is consistently significant, which verifies the robustness of our method to fit complex distributions.



(a) MNIST (b) EMNIST

Fig. 6. Performance w.r.t data heterogeneity.

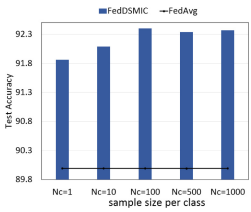
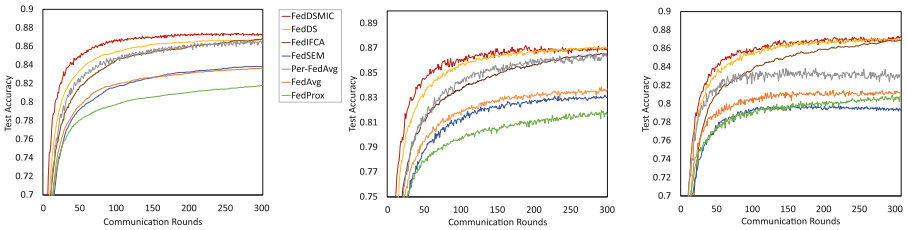


Fig. 7. Effects of indicator samples on MNIST with $\alpha = 0.5$.

Effects of the Indicator Sample Size. We test the effect of FedDSMIC with different indicator sample sizes on MNIST as in Fig. 7. N_c is the number of samples per class. The gain of FedDSMIC over FedAvg is consistently remarkable given different indicator sample sizes, whereas a sufficient number of indicator samples brings better performance. The model accuracy is high enough when $N_c = 1$, proving that it is acceptable to distinguish different customers despite the small sample size. When N_c reaches 100, the performance tends to stabilize.

Impacts of Straggler Clients. We explore different numbers of total clients versus active clients on FEMNIST, with active ratios of 0.2, 0.5, and 1.0, respectively. Figure 8 that FedDSMIC consistently outperforms all baselines under various active client settings. Combined with Fig. 8(a),(b), and (c), we can observe that our approach requires much fewer communication rounds to reach the same performance, regardless of the setting of straggler clients.



(a) r = 1.0 (b) r = 0.5 (c) r = 0.2

Fig. 8. Average test accuracy vs. communication rounds on FEMNIST w.r.t different ratio of straggler clients

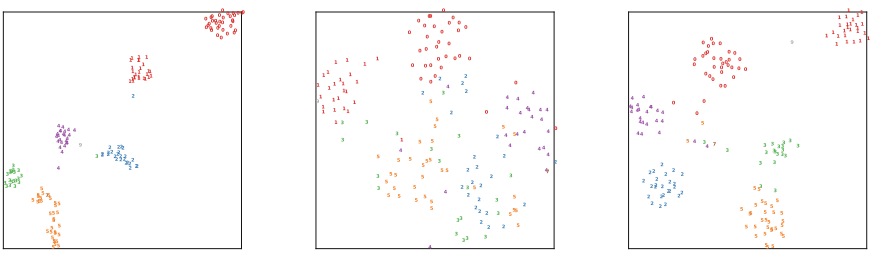
Effects of Communication Frequency: We explore local updating steps τ and local epochs E for CIFAR10 and CIFAR100, respectively. The higher τ or E means longer communication delays before global communication. As shown in the last two rows of Table 2, our approach is robust against different levels of communication delays.

5.5 Ablation Results

We set up ablation experiments to explore the role of the two points. FedDS (clustering-only of our method) and FedDSMIC are shown in the last two columns of Table 2. The results indicate that FedDS is already better than other cluster FL methods, especially the faster convergence speed (Fig. 4 and Fig. 8), which demonstrates our similarity calculation method detects the data distribution to cluster accurately. FedDSMIC can further improve performance by building personalized models for clients, especially under complicated data distribution.

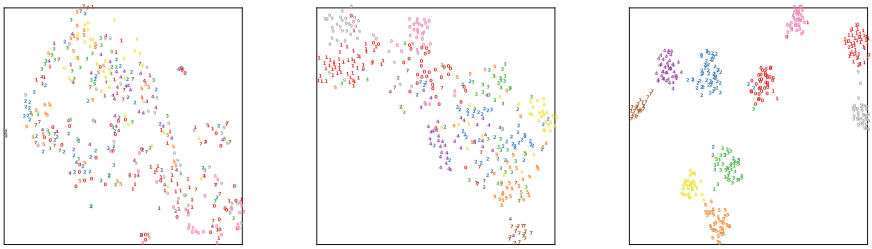
5.6 Discussion

Finally, we visualize the feature representations of the training data via client-side models on CIFAR10 obtained by Local, FedAvg, and FedDSMIC, respectively. There are two types of clients. One is shown in Fig. 9, which has enough local data to train its model. The performance of FedAvg degrades because the global model does not fit the local data distribution. In contrast, the client model obtained with FedDSMIC can benefit clients with similar data distributions. Therefore, the feature representation of the client model is looser than the local-trained model under the premise of ensuring good classification ability, i.e., the model's generalization will be more robust. The other is shown in Fig. 10, whose local data is insufficient to train a good model. Although FedAvg performs better than Local, the accuracy is still low due to heterogeneous data distribution. Whereas FedDSMIC can get an excellent personalized model that fits the data distribution nearly perfectly, leading to better generalization.



(a) LocalTrain (b) FedAvg (c) Ours

Fig. 9. The visualization of feature representations of 59-th local model with t-SNE for CIFAR10.



(a) LocalTrain (b) FedAvg (c) Ours

Fig. 10. The visualization of feature representations of 17-th local model with t-SNE for CIFAR10.

6 Conclusion

In this paper, we proposed a new clustered multi-task framework based on the assumption of a realistic cluster structure to address client heterogeneity. Our algorithm detected the local underlying data distribution by the trained model's memory to cluster similar clients. Furthermore, we learned personalized models for each client based on the initial cluster model to address the limitations of existing clustered FL. Extensive empirical evaluation has shown that our approach trained models with higher accuracy, fairness, and lower resource consumption than state-of-the-art FL algorithms, even for clients not present at training time.

Acknowledgments. This work is supported by The National Key Research and Development Program of China No. 2021YFB3101400 and the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDC02040400.

References

1. Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Vaughan, J.W.: A theory of learning from different domains. Mach. Learn. **79**(1), 151–175 (2010)
2. Bezdek, J.C., Hathaway, R.J.: Some notes on alternating optimization. In: Pal, N.R., Sugeno, M. (eds.) AFSS 2002. LNCS (LNAI), vol. 2275, pp. 288–300. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45631-7_39
3. 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. IEEE (2020)
4. Caldarola, D., Mancini, M., Galasso, F., Ciccone, M., Rodolà, E., Caputo, B.: Cluster-driven graph federated learning over multiple domains. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2749–2758 (2021)
5. Caldas, S., et al.: Leaf: a benchmark for federated settings. arXiv preprint arXiv:1812.01097 (2018)

6. Chen, F., Luo, M., Dong, Z., Li, Z., He, X.: Federated meta-learning with fast convergence and efficient communication. arXiv preprint arXiv:1802.07876 (2018)
7. Cohen, G., Afshar, S., Tapson, J., Van Schaik, A.: EMNIST: extending MNIST to handwritten letters. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 2921–2926. IEEE (2017)
8. Corinzia, L., Beuret, A., Buhmann, J.M.: Variational federated multi-task learning. arXiv preprint arXiv:1906.06268 (2019)
9. Cortes, C., Mohri, M.: Domain adaptation and sample bias correction theory and algorithm for regression. Theoret. Comput. Sci. **519**, 103–126 (2014)
10. Fallah, A., Mokhtari, A., Ozdaglar, A.: On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In: International Conference on Artificial Intelligence and Statistics, pp. 1082–1092. PMLR (2020)
11. Fallah, A., Mokhtari, A., Ozdaglar, A.: Personalized federated learning with theoretical guarantees: a model-agnostic meta-learning approach. Adv. Neural. Inf. Process. Syst. **33**, 3557–3568 (2020)
12. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning, pp. 1126–1135. PMLR (2017)
13. French, R.M.: Catastrophic forgetting in connectionist networks. Trends Cogn. Sci. **3**(4), 128–135 (1999)
14. Ghosh, A., Chung, J., Yin, D., Ramchandran, K.: An efficient framework for clustered federated learning. Adv. Neural Inf. Process. Syst. **33**, 19586–19597 (2020)
15. Hanzely, F., Richtárik, P.: Federated learning of a mixture of global and local models. arXiv preprint arXiv:2002.05516 (2020)
16. Hsieh, K., Phanishayee, A., Mutlu, O., Gibbons, P.: The non-IID data quagmire of decentralized machine learning. In: International Conference on Machine Learning, pp. 4387–4398. PMLR (2020)
17. Huang, Y., et al.: Personalized cross-silo federated learning on non-IID data. In: AAAI, pp. 7865–7873 (2021)
18. Jiang, Y., Konečnỳ, J., Rush, K., Kannan, S.: Improving federated learning personalization via model agnostic meta learning. arXiv preprint arXiv:1909.12488 (2019)
19. Kairouz, P., et al.: Advances and open problems in federated learning. Found. Trends® Mach. Learn. **14**(1–2), 1–210 (2021)
20. Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S., Suresh, A.T.: Scaffold: stochastic controlled averaging for federated learning. In: International Conference on Machine Learning, pp. 5132–5143. PMLR (2020)
21. Khodak, M., Balcan, M.F.F., Talwalkar, A.S.: Adaptive gradient-based meta-learning methods. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
22. Konečnỳ, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016)
23. Krizhevsky, A.: Learning multiple layers of features from tiny images. Master's thesis, University of Tront (2009)
24. Kulkarni, V., Kulkarni, M., Pant, A.: Survey of personalization techniques for federated learning. In: 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), pp. 794–797. IEEE (2020)

25. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
26. Li, T., Hu, S., Beirami, A., Smith, V.: Ditto: fair and robust federated learning through personalization. In: International Conference on Machine Learning, pp. 6357–6368. PMLR (2021)
27. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: challenges, methods, and future directions. IEEE Sig. Process. Mag. **37**(3), 50–60 (2020)
28. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. Proc. Mach. Learn. Syst. **2**, 429–450 (2020)
29. Mansour, Y., Mohri, M., Ro, J., Suresh, A.T.: Three approaches for personalization with applications to federated learning. arXiv preprint arXiv:2002.10619 (2020)
30. Mansour, Y., Mohri, M., Rostamizadeh, A.: Domain adaptation: learning bounds and algorithms. arXiv preprint arXiv:0902.3430 (2009)
31. Marfoq, O., Neglia, G., Bellet, A., Kameni, L., Vidal, R.: Federated multi-task learning under a mixture of distributions. In: Advances in Neural Information Processing Systems, vol. 34 (2021)
32. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: the sequential learning problem. In: Psychology of Learning and Motivation, vol. 24, pp. 109–165. Elsevier (1989)
33. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics, pp. 1273–1282. PMLR (2017)
34. Mohri, M., Sivek, G., Suresh, A.T.: Agnostic federated learning. In: International Conference on Machine Learning, pp. 4615–4625. PMLR (2019)
35. Nayak, G.K., Mopuri, K.R., Shaj, V., Radhakrishnan, V.B., Chakraborty, A.: Zero-shot knowledge distillation in deep networks. In: International Conference on Machine Learning, pp. 4743–4751. PMLR (2019)
36. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv 2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)
37. Sarkar, S., Ghosh, A.K.: On perfect clustering of high dimension, low sample size data. IEEE Trans. Pattern Anal. Mach. Intell. **42**(9), 2257–2272 (2019)
38. Sattler, F., Müller, K.R., Samek, W.: Clustered federated learning: model-agnostic distributed multitask optimization under privacy constraints. IEEE Trans. Neural Netw. Learn. Syst. **32**(8), 3710–3722 (2020)
39. Smith, V., Chiang, C.K., Sanjabi, M., Talwalkar, A.S.: Federated multi-task learning. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
40. Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., Khazaeni, Y.: Federated learning with matched averaging. arXiv preprint arXiv:2002.06440 (2020)
41. Wang, K., Mathews, R., Kiddon, C., Eichner, H., Beaufays, F., Ramage, D.: Federated evaluation of on-device personalization. arXiv preprint arXiv:1910.10252 (2019)
42. Xie, M., et al.: Multi-center federated learning. arXiv preprint arXiv:2108.08647 (2021)
43. Zhang, Y., Xiang, T., Hospedales, T.M., Lu, H.: Deep mutual learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4320–4328 (2018)
44. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-IID data. arXiv preprint arXiv:1806.00582 (2018)

45. Zhou, Z., Chu, L., Liu, C., Wang, L., Pei, J., Zhang, Y.: Towards fair federated learning. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 4100–4101 (2021)
46. Zhu, Z., Hong, J., Zhou, J.: Data-free knowledge distillation for heterogeneous federated learning. In: International Conference on Machine Learning, pp. 12878–12889. PMLR (2021)