

# Rethinking Personalized Federated Learning with Clustering-based Dynamic Graph Propagation

Jiaqi Wang<sup>1\*</sup>, Yuzhong Chen<sup>2</sup>, Yuhang Wu<sup>2</sup>, Mahashweta Das<sup>2</sup>, Hao Yang<sup>2</sup>, and Fenglong Ma<sup>1</sup>

<sup>1</sup> The Pennsylvania State University {jqwang, fenglong}@psu.edu

<sup>2</sup> Visa Research {yuzchen, yuhawu, mehdas, haoyang}@psu.edu

**Abstract.** Most existing personalized federated learning approaches are based on intricate designs, which often require complex implementation and tuning. In order to address this limitation, we propose a simple yet effective personalized federated learning framework. Specifically, during each communication round, we group clients into multiple clusters based on their model training status and data distribution on the server side. We then consider each cluster center as a node equipped with model parameters and construct a graph that connects these nodes using weighted edges. Additionally, we update the model parameters at each node by propagating information across the entire graph. Subsequently, we design a precise personalized model distribution strategy to allow clients to obtain the most suitable model from the server side. We conduct experiments on three image benchmark datasets and create synthetic structured datasets with three types of typologies. Experimental results demonstrate the effectiveness of the proposed FEDCEDAR.

**Keywords:** Federated learning, Model personalization and distribution

## 1 Introduction

Federated learning (FL) [18] enables different data holders to cooperatively train machine learning models without sharing data. However, data heterogeneity poses a significant challenge in FL. To tackle this issue, personalized FL (PFL) has been proposed. Among the various research tracks in PFL, local model personalization through parameter decoupling [21] and clustering [5, 27] have been explored. Parameter decoupling-based PFL has several drawbacks: (1) It demands intensive computational resources and is complex to implement. (2) It encounters the challenge of dividing the private or federated parameters [1]. In contrast, clustering-based approaches do not have such additional requirements or limitations. However, existing clustering-based approaches overlook the hidden relations between clusters and may require additional designs to handle the intricate mechanism [2], which complicates implementation and deployment on other frameworks. Consequently, a challenging yet practical question arises: *Is it possible to design a **simple** yet **effective** PFL framework that can address the data heterogeneity problem while considering the capture of hidden relations across clients?*

---

\* This work was done when Jiaqi Wang interned at Visa.

To address this question, we encounter several non-trivial challenges: **C1: Hidden relation capturing.** In real-world applications, there may exist a physical topology among clients. Utilizing this topology effectively would contribute to local model updating. However, clients are unable to share data or access global topology information. Accurately capturing the hidden relations among clients becomes a challenging task. **C2: Clustering-based knowledge sharing.** In FL frameworks, a random subset of clients participates in iterative model learning. Learning client-level hidden relations can be inefficient. Also, propagating under-trained models among all clients can lead to failure in FL model training. Clustering approaches allow similar clients to cooperate, which helps avoid such issues. However, most research focuses solely on gathering knowledge within clusters, disregarding information across multiple clusters. Extracting appropriate knowledge to facilitate model updates across all clusters becomes an urgent matter to address. **C3: Fitted model acquisition.** Distributing the most fitting models back to clients as their initialization in the next communication round is not trivial. Designing an accurate model distribution strategy is crucial to maintain and inherit the benefits of iterative training, maximizing the utilization of model updates and effectively supporting model training in subsequent rounds. Moreover, the proposed approach should be easy to implement without requiring specific or complicated tuning.

To address all the aforementioned challenges, we propose a simple yet effective personalized federated learning framework, denoted as FEDCEDAR. The framework is shown in Figure 1. In our approach, we first perform local model training using the respective client’s data and upload the model parameters to the server. On the server side, we cluster the collected models into different groups based on their parameters. To enhance our framework, we propose a method incorporating dynamic graph construction and weighted knowledge propagation. Each cluster center is treated as a node with associated model parameters, and a graph is constructed connecting these nodes with weighted edges. Subsequently, the model parameters stored at each node are updated across the entire graph by leveraging information from other nodes through the weighted edges. Throughout the federated learning iterations, the active clients may vary, causing the clusters and graphs to change dynamically in each communication round. Finally, we design a simple yet precise personalized model distribution strategy to maintain the benefits of iterative training and further enhance model personalization. Depending on whether an active client was selected in the previous update iteration, we assign either the personalized cluster center models or the aggregated ones to the clients.

## 2 Related Work

FL is initially proposed in [18], which proposes a classical algorithm named FedAvg. Till this research, FL has recently been explored in multiple directions[16, 24, 20, 17, 10, 23, 12, 22, 11, 13]. Considering our proposed work, we discuss related works of model personalization and graph in FL in the following. **Personalized Federated Learning:** Personalized FL cares more about each local model’s performance, which is more sufficient and practical under the non-IID setting. One research track to solve this problem is named FedAvg+, which focuses on how to balance the global aggregation and local training[9, 4, 21]. Besides the research track that we discussed, some research

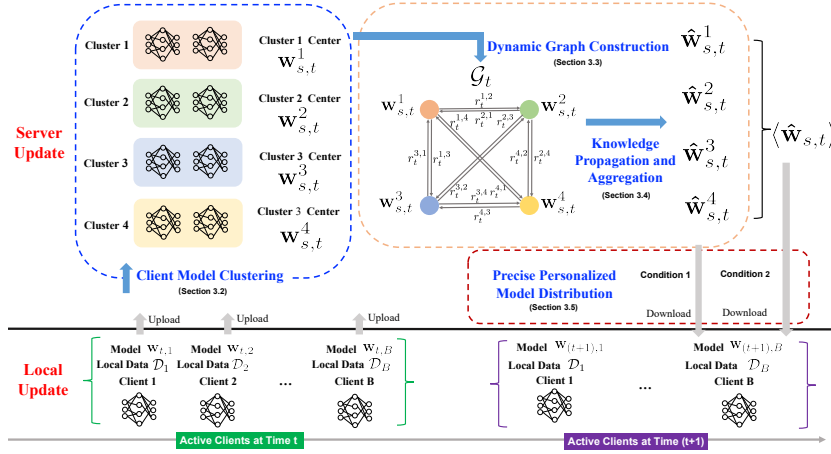


Fig. 1: Overview of the proposed FEDCEDAR ( $K = 4$  as an example). Note that clients selected at time  $t$  may be different from those that are selected at time  $t + 1$ . **Condition 1** means that the clients are selected at time  $t$  as well as  $(t + 1)$ , and **Condition 2** means that the clients are not selected at time  $t$ .

works focus on parameter decoupling [21] and clustering [5, 27] to conduct local model personalization in FL. However, all discussed personalization FL works ignore the hidden relations between the clients or the clusters. **Graph in Federated Learning:** Most research works, which get the graph involved in FL, focus on training GNN (Graph Neural Network) models with graph structure data distributedly [26, 6, 28, 7, 15]. In a recent work [3], authors treat each local client as a node to conduct GCN learning at the server. But the number of models maintained at the server is the same as the client number, which would take intensive computation resources. Besides that, the adjacency matrix is required to be known or learned in their proposed work, which affects its practicability.

### 3 Methodology

#### 3.1 Model Overview

Figure 1 shows the overview of the proposed framework, which consists of two main modules, i.e., *local update* and *server update*. In the **local update** step, we first train a local model  $w_n$  with local data  $\mathcal{D}_n$ , where  $\mathcal{D}_n = \{(x_i^n, y_i^n)\}_{i=1}^{H_n}$  and  $H_n$  is the number of data examples at client  $n$ . Specifically, we use the cross entropy (CE) loss to train  $w_n$  as follows:  $\mathcal{L}_n = \text{CE}(f(\mathbf{x}^n; w_n), \mathbf{y}^n)$ , where  $f(\cdot; \cdot)$  represents the neural network,  $\mathbf{x}^n$  represents the client data representations, and  $\mathbf{y}^n$  is the corresponding label vector. Following the training procedure of general federated learning models such as FedAvg [18], a small subset of clients with size  $B$  will be randomly selected to conduct the local model learning at the  $t$ -th communication round. The outputs from the local client update are a set of trained model parameters  $\{w_{t,1}, w_{t,2}, \dots, w_{t,B}\}$ , which will be uploaded to the server for the server update.

There are four critical components including *client model clustering*, *dynamic graph construction*, *knowledge propagation*, and *precise personalized model distribution*. After receiving  $\{\mathbf{w}_{t,1}, \mathbf{w}_{t,2}, \dots, \mathbf{w}_{t,B}\}$ , FEDCEDAR will conduct the client model clustering using the K-means algorithm, i.e., dividing the  $B$  uploaded clients into  $K$  clusters, where  $K$  is the predefined number of clusters. The outputs of this step are a set of cluster centers  $\mathcal{V}_t = \{\mathbf{w}_{s,t}^2, \mathbf{w}_{s,t}^1, \dots, \mathbf{w}_{s,t}^K\}$ , where  $\mathbf{w}_{s,t}^k$  denotes the center of the  $k$ -th cluster at the  $t$ -th communication round. In the dynamic graph construction step, FEDCEDAR treats each cluster center as a node and constructs a dynamic weighted graph  $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{R}_t)$ , where  $\mathcal{E}_t \subseteq \mathcal{V}_t \times \mathcal{V}_t$  is the set of edges between nodes, and  $\mathcal{R}_t$  is the set of weights on edges. FEDCEDAR then executes the knowledge propagation step across the whole graph  $\mathcal{G}_t$  to update node representations, i.e., model personalization learning. Finally, FEDCEDAR distributes the learned personalized models  $\{\hat{\mathbf{w}}_{s,1}^1, \hat{\mathbf{w}}_{s,1}^2, \dots, \hat{\mathbf{w}}_{s,t}^K\}$  back to the corresponding clients with the precise model distribution strategy. Next, we describe the details of the four components in the server update step.

### 3.2 Client Model Clustering

At the  $t$ -th communication round, the  $B$  selected clients will upload their models  $\{\mathbf{w}_{t,1}, \dots, \mathbf{w}_{t,B}\}$  to the server. These models will be grouped into  $K$  clusters using K-means [14][5] by optimizing the following loss function:

$$J_t = \sum_{k=1}^K \sum_{b=1}^B \delta_{t,b}^k (\|\mathbf{w}_{s,t}^k - \mathbf{w}_{t,b}\|^2), \quad (1)$$

where  $\|\mathbf{w}_{s,t}^k - \mathbf{w}_{t,b}\|^2$  is the Euclidean distance between the  $k$ -th center  $\mathbf{w}_{s,t}^k$  and the client model  $\mathbf{w}_{t,b}$ .  $\delta_{t,b}^k$  is the indicator. If the  $b$ -th client belongs to the  $k$ -th cluster, then  $\delta_{t,b}^k = 1$ . Otherwise,  $\delta_{t,b}^k = 0$ . Note that for each communication round  $t$ , we will generate a new set of  $K$  cluster centers  $\{\mathbf{w}_{s,t}^1, \dots, \mathbf{w}_{s,t}^K\}$ .

### 3.3 Dynamic Weighted Graph Construction

The clustering process helps local clients with similar data distributions or characteristics to aggregate together based on their model parameters. However, simply utilizing the results of the clustering algorithm to conduct the personalization in FL [27, 5] has several limitations. On the one hand, the cluster centers  $\{\mathbf{w}_{s,t}^1, \dots, \mathbf{w}_{s,t}^K\}$  are obtained by averaging the client parameters within each cluster. In other words, the personalization information is only from the clients within each cluster by modeling the *inner-cluster* characteristics, which ignores that *between clusters*. On the other hand, the performance of existing clustering approaches is mainly determined by the quality of the cluster assignment. One low-quality cluster may lead to slow convergence and poor overall performance. When modeling the relations between clusters, high-quality information can flow to other clusters, which may neutralize the negative effect of low-quality clusters. Thus, it is essential to model the hidden relations between clusters.

Towards this end, we build a dynamic weighted graph to capture the hidden relations between the local models to enhance the model updates further. However, it is

highly non-trivial to design a novel mechanism satisfying the following conditions and requirements. First, in the FL setting, it would be better not to increase the network transmission load due to the constraints of communication cost. Second, in a typical FL framework, we randomly sample clients at each communication round, which means different batches of clients contribute to the updates. Then the center model  $\mathbf{w}_{s,t}^k$  in each cluster carries different information with huge differences. How to leverage the iterative heterogeneous knowledge appropriately is a challenge. FedAvg-based aggregation approach [18] enables rough information sharing by obtaining one global model, which cannot handle the data or model heterogeneity. Thus, solving such challenges requires a more reliable and convincing aggregation strategy to maintain the personalization property for each local client.

Specifically, we use  $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{R}_t)$  to denote a graph, where  $\mathcal{V}_t = \{\mathbf{w}_{s,t}^1, \dots, \mathbf{w}_{s,t}^K\}$  is the node set, and each node corresponds to a cluster.  $\mathcal{E}_t \subseteq \mathcal{V}_t \times \mathcal{V}_t$  is the set of edges, where any pair of nodes exists an edge with a corresponding weight.  $\mathcal{R}_t$  is the set of edge weights. The weight is defined as the normalized cosine similarity between two cluster centers  $\mathbf{w}_{s,t}^i$  and  $\mathbf{w}_{s,t}^j$ , where  $\sum_{j=1}^K r_t^{i,j} = 1$ .

### 3.4 Knowledge Propagation and Aggregation

After we build the graph with the weighted edges, we are able to reveal and describe the hidden relations between cluster centers corresponding to a set of model parameters. The next question is how we could utilize the connection to help the model update across the graph appropriately. To answer this question, we design an effective approach to enable the model parameters saved at each node to gather information from other nodes for knowledge propagation across the whole graph.

Mathematically, given a constructed graph  $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{R}_t)$ , for  $\forall k \in K$ , we update the center in a weighted sum way as follows:

$$\begin{aligned} \hat{\mathbf{w}}_{s,t}^k &= g([\mathbf{w}_{s,t}^{1,P-1}, \dots, \mathbf{w}_{s,t}^{K,P-1}], [r_t^{k,1}, \dots, r_t^{k,K}], P) = \sum_{i=1}^K r_t^{k,i} \mathbf{w}_{s,t}^{i,P-1} \\ &= \sum_{i=1}^K r_t^{k,i} \sum_{j=1}^K r_t^{i,j} \mathbf{w}_{s,t}^{j,P-2} = \dots = \underbrace{\sum_{i=1}^K r_t^{k,i} \sum_{j=1}^K r_t^{i,j} \dots \sum_{z=1}^K r_t^{u,z} \mathbf{w}_{s,t}^{z,0}}_{k \leftarrow i \leftarrow j \leftarrow \dots \leftarrow u \leftarrow z (\# \text{ path} = P)} \end{aligned} \quad (2)$$

where  $g$  is the function to conduct the knowledge propagation with the updated model parameters, and  $P$  denotes the times we repeat the propagation process.  $\mathbf{w}_{s,t}^{i,P-1}$  is the updated model parameter of node  $i$  after  $P-1$  rounds of propagation and aggregation,  $r_{k,i}$  is the weight between node  $k$  and  $i$ . When  $P=1$ ,  $\mathbf{w}_{s,t}^{z,0} = \mathbf{w}_{s,t}^z$ , which is the output from Section 3.3. After this procedure, we have a set of new models named  $\{\hat{\mathbf{w}}_{s,t}^1, \hat{\mathbf{w}}_{s,t}^2, \dots, \hat{\mathbf{w}}_{s,t}^K\}$  with the weighted information from in-cluster clients and out-of-cluster nodes.

### 3.5 Precise Personalized Model Distribution

After we get  $K$  customized models at the server side, it is challenging to select and distribute the most fitted model to each individual active client to achieve the optimal local

personalization performance at the next communication round. There are two reasons: (1) To simulate the real-world scenario, we randomly sample clients at each communication round with the sample ratio  $\gamma$ . In this case, each sampled client set could vary a lot. (2) We still expect to maintain the model generalization and personalization with the benefit of the iterative training mechanism in FL. To solve the above challenges, after obtaining  $\{\hat{\mathbf{w}}_{s,t}^1, \hat{\mathbf{w}}_{s,t}^2, \dots, \hat{\mathbf{w}}_{s,t}^K\}$ , we design a precise personalized model distribution strategy.

Specifically, we need to consider two conditions with respect to the client sampling at two consecutive communication rounds  $t-1$  and  $t$ . Correspondingly, there are sampled client set named  $\mathcal{C}_{t-1}$  and  $\mathcal{C}_t$  at time  $t-1$  and  $t$ , respectively. Given  $\mathcal{C}_{t-1}$  and  $\mathcal{C}_t$ , for client  $n$  at communication round  $t$ , different model distribution strategies are implemented under the following two conditions:

- **Condition 1:** if  $n \in \mathcal{C}^* = \mathcal{C}_{t-1} \cap \mathcal{C}_t$  (*intersection set*) at time  $t$ , we trace back to the cluster  $k$ , to which client  $n$  belongs at time  $t-1$ . Then we distribute model  $\hat{\mathbf{w}}_{s,t-1}^k$  to client  $n$  as the model initialization at time  $t$ ;
- **Condition 2:** if  $n \in \mathcal{C}' = \mathcal{C}_t - \mathcal{C}_{t-1}$  (*difference set*) at time  $t-1$ , we distribute  $\langle \hat{\mathbf{w}}_{s,t-1} \rangle$  to client  $n$  as the initial model parameters, where  $\langle \hat{\mathbf{w}}_{s,t-1} \rangle$  is calculated as follows:  $\langle \hat{\mathbf{w}}_{s,t-1} \rangle = \frac{1}{K}(\hat{\mathbf{w}}_{s,t-1}^1 + \hat{\mathbf{w}}_{s,t-1}^2 + \dots + \hat{\mathbf{w}}_{s,t-1}^K)$ .

## 4 Experiment

### 4.1 Experiment Setup

**Dataset Preparation** We conduct experiments for the image classification task on **MNIST**, **SVHN**, and **CIFAR-10** datasets in both IID and non-IID data distribution settings, respectively. We split the training datasets into 80% for training and 20% for testing. For the IID setting, the training and testing datasets are both randomly sampled. For the non-IID setting, we divide the training dataset following the approach used in [3] and set the *shard* = 2, which is an extreme non-IID setting. To test the personalization effectiveness, we sample the testing dataset following the label distribution as the training dataset.

**Baselines** The proposed FEDCEDAR is a personalized federated learning algorithm. To achieve personalization, we adopt the clustering technique and construct dynamically weighted graphs. To fairly evaluate the proposed FEDCEDAR, we use the following baselines: (1) *Classical FL Models*: **FedAvg** [18] and **FedProx** [9]; (2) *Personalized FL Models*: **pFedMe** [21] and **pFedBayes** [29]; (3) *Graph-based FL Model*: **SFL** [3]; (4) *Clustering-based FL Models*: **IFCA** [5] and **FedSem**[27].

**Implementation Details** In our experiments, we leverage convolutional neural networks (CNNs) as our basic models for the three image datasets. For the MNIST and SVHN datasets, the network structure consists of three convolutional layers and two fully-connected layers. For the CIFAR-10 dataset, we have six convolutional layers and two fully-connected layers. All the baselines and FEDCEDAR use the same networks to keep the comparison fair. Following [25], the total client number is  $N = 100$ , and the

Table 1: Performance comparison with baselines.

Category	Dataset Setting	MNIST		SVHN		CIFAR-10	
		IID	non-IID	IID	non-IID	IID	non-IID
Classical	FedAvg	96.77%	91.02%	82.65%	81.04%	68.93%	59.85%
	FedProx	97.87%	93.98%	83.09%	82.68%	71.07%	64.07%
Personalization	Per-FedAvg	96.57%	<u>93.56%</u>	87.74%	87.09%	71.32%	75.56%
	pFedMe	96.90%	93.10%	87.79%	86.37%	73.92%	77.21%
	pFedBayes	97.23%	92.08%	<u>90.69%</u>	<u>88.03%</u>	<u>80.45%</u>	<u>79.05%</u>
Graph	SFL	96.88%	93.10%	86.97%	86.15%	79.92%	75.44%
Cluster	IFCA	<b>97.96%</b>	92.09%	84.44%	83.26%	79.21%	73.08%
	FedSem	96.52%	93.05%	84.75%	84.96%	79.87%	71.22%
Ours	FEDCEDAR	<u>97.90%</u>	<b>95.96%</b>	<b>91.19%</b>	<b>88.76%</b>	<b>82.80%</b>	<b>81.53%</b>

active client ratio in each communication round is 30%. The total communication round between the server and local clients is  $T = 100$ . The local training batch size is 16, the local training epoch is 5, and the local training learning rate is 0.01. The number of knowledge propagation  $P$  is set to 2. The cluster number  $K$  is set to 5. We provide the hyperparameter study to explore how the key hyperparameters affect the performance of the proposed FEDCEDAR in the subsection 4.5.

## 4.2 Performance Evaluation

We run each approach three times and report the *average accuracy* in Table 1. There are several observations and discussions as below: (1) Overall, our proposed FEDCEDAR outperforms baselines on MNIST, SVHN, and CIFAR-10 datasets under both IID and non-IID settings except for the result of IFCA on MNIST under the IID setting. This is due to the fact that the MNIST dataset under the IID setting is a relatively easy task where all approaches achieve comparable performance, compared with other datasets and settings; (2) If we focus on SVHN and CIFAR-10, we find that the advantage of our algorithm is becoming dominant, with the dataset being more complicated under the non-IID setting. Compared with FedAvg under the non-IID setting, our approach increases the accuracy rate by 9.53% and 36.22%, respectively. Compared with the best performance of other algorithms on SVHN (88.03%) and CIFAR-10 (79.05%) under the non-IID setting, our algorithm boosts the performance to 88.76% ( $\uparrow 0.73\%$ ) and 81.53% ( $\uparrow 2.48\%$ ), respectively; (3) Among all baselines, pFedBayes performs the best on both SVHN and CIFAR-10 datasets. pFedBayes is specifically tailored for personalization. However, it requires the exchange of data distribution information between the clients and the server, which raises concerns about communication costs and privacy leakage.

## 4.3 Ablation Study

In this subsection, we conduct the ablation study to investigate the contribution of key modules in our proposed model FEDCEDAR. We use the following reduced or modified

models as baselines: **AS-1: Without clustering.** We treat each client as a node to build the graph and conduct the knowledge propagation. Compared with [3], we do not have the complicated GCN learning process or the optimal global model but distribute its own model via our precise model distribution strategy. **AS-2: Without building a graph or knowledge propagation.** We only conduct client clustering, and other modules are kept. Different from [5], we only conduct one clustering process rather than an iterative optimization to estimate the cluster identities. **AS-3: Without precise personalized model distribution.** After conducting dynamic graph construction and knowledge propagation, we aggregate the models from each node into one single model and distribute it back to the clients.

Table 2 shows the experimental results, where we can observe that reducing one or more key modules in FEDCEDAR leads to performance degradation. There are several observations and discussions as below: (1) The results compared with AS-1 show that treating each client as a node in the graph may not be able to extract enough supporting knowledge from other clients. It also shows that the help of similar clients within one cluster can enhance the model performance; (2) In AS-2, we construct the clusters and maintain the precise personalized distribution strategy. The reduction of the performance can tell the effectiveness of utilizing the hidden relations between each cluster via graph construction and knowledge propagation. It reveals that the extra weighted knowledge is able to lift the model performance; (3) The performance of AS-3 is the worst compared with AS-1 and AS-2. If we directly aggregate all the models at the nodes into one global model via doing an average, it essentially can be simplified as FedAvg.

Table 2: Ablation study.

Dataset Setting	MNIST		SVHN		CIFAR-10	
	IID	non-IID	IID	non-IID	IID	non-IID
AS-1	96.79%	92.16%	82.66%	82.11%	70.10%	72.96%
AS-2	96.80%	92.53%	82.71%	81.39%	70.45%	69.15%
AS-3	97.04%	92.01%	83.66%	81.05%	71.08%	65.37%
FEDCEDAR	<b>97.90%</b>	<b>95.96%</b>	<b>91.19%</b>	<b>88.76%</b>	<b>82.80%</b>	<b>81.53%</b>

#### 4.4 Case Study

To further explore the effectiveness of the proposed FEDCEDAR, we design a case study to demonstrate the process of the graph formulation with respect to the communication round. We are interested in testing if our model is able to capture and reconstruct the hidden relations between local clients through iterative updates in FEDCEDAR framework.

**Structured Data Construction** To achieve our target, we structure a synthetic dataset via MNIST by building known relations between the nodes shown in Figure 2, and the goal is to test whether our graph construction method has the capability of identifying these pre-designed relations. For each node in the graph, we have 20 clients equipped with designed label distributions. If there are common labels among a pair of nodes, an edge connects them. For example, in Topology 1, label 2 (denoted as L2) is the common label between node 1 and node 2, so there is an edge connecting the two nodes. In this case study, we construct three topologies with 3 nodes, 4 nodes, and 5 nodes, respectively.



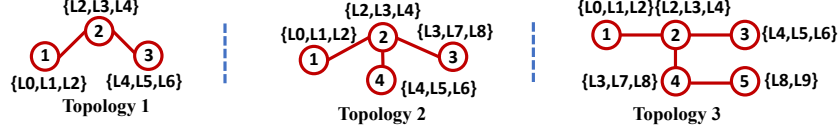


Fig. 2: Structured datasets with synthetic topologies.

**Periodic Activation Strategy** For each node in the graph, we have 20 clients, where each client has the corresponding labels from the node as we discussed. Given a periodic round set  $\mathcal{Q}$ , if  $t \in \mathcal{Q}$ , we sample the clients from each node at the activation ratio  $\gamma$  to formulate the active client set, which guarantees the formulated client set covers the label information from all the structured nodes in the graph. At other communication rounds  $t \notin \mathcal{Q}$ , we randomly sample  $B$  clients from  $N$  clients without considering where the clients come from, where  $B = N * \gamma$ . Thus, the total number of active clients is defined in this simulation experiment as follows: if  $t \in \mathcal{Q}$ :  $\mathcal{C}_t = \sum_{i=1}^K (S_i * \gamma)$ ; if  $t \notin \mathcal{Q}$ :  $\mathcal{C}_t = B$ , where  $S_i$  is the client set at node  $i$ . Later, we track the graph construction performance at the round set  $\mathcal{Q}$  through the communication rounds compared with the synthetic topology structure.

**Evaluation Metrics** We leverage the random index [19] as the evaluation metric to represent the performance of the constructed clusters across the iterative update process. It measures the similarity of the two clustering outcomes by considering all pairs of samples: it counts pairs that are assigned in the same or different clusters in the generated and ground-truth clusters [8]. Specifically in our case, given a set  $\{\hat{\mathbf{w}}_{s,t}^1, \hat{\mathbf{w}}_{s,t}^2, \dots, \hat{\mathbf{w}}_{s,t}^K\}$ , assuming we have two partitions  $\mathcal{A} = \{A_1, A_2, \dots, A_a\}$  and  $\mathcal{B} = \{B_1, B_2, \dots, B_b\}$ , where there are  $a$  and  $b$  subsets in each partition respectively, we have:  $\mathbf{R} = (\alpha + \beta) / \binom{K}{2} = 2 \times (\alpha + \beta) / K(K - 1)$ , where  $\alpha$  and  $\beta$  are the number of agreements between the two partitions that the pairs belong to the same subsets or different subsets. Intuitively, it tells the occurrence or the probability that the partitions agree on the data sample pairs. Thus, the score  $\mathbf{R} = 1$  means the perfect clustering, and  $\mathbf{R} = 0$  means poor clustering.

**Results Analysis** We run the case study with different cluster numbers  $K = 3, 4$ , and  $5$ . We choose to activate the targeted clients from the designed topology every 5 communication rounds, which means we have 40 data points during the total 200 communication rounds ( $\mathcal{Q} = \{5, 10, \dots, 200\}$ ). To have a better visualization, we plot the data with the scatter every 2 data points, which means we totally have 20 markers along with each line.

The results are shown in Figure 3. In the figure, the black, blue, and red demonstrate the results with the number of clusters equal to 3, 4, and 5, respectively. When  $K = 3$ , which is Topology 1, we can observe that there is very limited fluctuation at the first initial

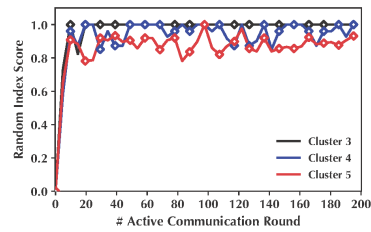


Fig. 3: Random index.

communication rounds. The lowest score is 0.73 and the score converges to 1 after several updates. For  $K = 4$ , the convergence speed is slower than the setting when  $K = 3$ , but it is close to 1 finally. Besides that, when  $K = 5$ , its convergence speed and score are not as good as when  $K = 3$  or 4. There are several possible reasons: (1) the complexity of the topology will increase the difficulty of generating the correct clusters and graphs; (2) with more nodes in our setting, there are more labels and clients involved in the updates, which increases the heterogeneity of the datasets. Generally, we observe that the random index score under three settings all reach convergence and converge to a high value, demonstrating the effectiveness and interpretability of our proposed algorithm.

#### 4.5 Hyperparameter Study

In this subsection, we investigate how the hyperparameters  $P$  and  $K$  affect the performance of FEDCEDAR.  $P$  controls how many times we conduct knowledge propagation across the graph at each communication round in Eq. (2).  $K$  controls the number of clusters when we group the clients by optimizing Eq. (1). To explore the parameter sensitivity, we alter  $P$  and  $K$  as  $\{1, 2, 3, 4, 5\}$  and  $\{3, 4, 5, 6, 7\}$ , respectively.

We report the three-time average experiment results in Figure 4. For the experiment results on SVHN, we observe: (1) The highest accuracy is 88.83% appearing at  $\{P = 2, K = 4\}$ . (2) In general, with the increase of  $P$ , the accuracy shows non-monotonous behavior by firstly going up and then going down. For example, given  $K = 4, 5, 6$ , or 7, the best performance appears at the setting with  $P = 2$  or 3. (3) With the cluster number  $K$  becomes larger, appropriate increase of  $P$  is able to boost the performance. Specifically, when the cluster number  $K$  increase from 3 to 6, the best performance appears at the point when  $P = 1, 2$  and 3, respectively.

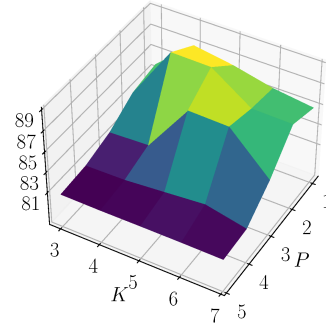


Fig. 4: Hyperparameter study.

## 5 Conclusion

We design a personalized federated learning framework FEDCEDAR equipped with the functionalities of clustering, dynamic weighted graph learning, and precise personalized model distribution to solve the data heterogeneity challenge. In particular, we build dynamic weighted graphs to conduct model aggregation and enable clients to obtain models precisely with our designed strategy. Our proposed FEDCEDAR outperforms state-of-the-art baselines on three datasets for the image classification task. To our best knowledge, this is the first FL research work to embed the clustering and dynamic weighted graph construction to explore the hidden relations between clients and thus obtain the optimal personalized local models via precise personalized model distribution strategies. In future works, we would like to provide theoretical convergence and generality analysis of our model under more flexible real-world settings.

## Bibliography

- [1] Arivazhagan, M.G., Aggarwal, V., Singh, A.K., Choudhary, S.: Federated learning with personalization layers. arXiv preprint arXiv:1912.00818 (2019)
- [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. pp. 1–9. IEEE (2020)
- [3] Chen, F., Long, G., Wu, Z., Zhou, T., Jiang, J.: Personalized federated learning with graph. arXiv preprint arXiv:2203.00829 (2022)
- [4] Fallah, A., Mokhtari, A., Ozdaglar, A.: Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems* **33**, 3557–3568 (2020)
- [5] Ghosh, A., Chung, J., Yin, D., Ramchandran, K.: An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems* **33**, 19586–19597 (2020)
- [6] He, C., Ceyani, E., Balasubramanian, K., Annavaram, M., Avestimehr, S.: Spreadgnn: Serverless multi-task federated learning for graph neural networks. arXiv preprint arXiv:2106.02743 (2021)
- [7] Hu, K., Wu, J., Li, Y., Lu, M., Weng, L., Xia, M.: Fedgcn: Federated learning-based graph convolutional networks for non-euclidean spatial data. *Mathematics* **10**(6), 1000 (2022)
- [8] Hubert, L., Arabie, P.: Comparing partitions. *Journal of classification* **2**(1), 193–218 (1985)
- [9] Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems* **2**, 429–450 (2020)
- [10] Lin, S., Han, Y., Li, X., Zhang, Z.: Personalized federated learning towards communication efficiency, robustness and fairness. *Advances in Neural Information Processing Systems* (2022)
- [11] Liu, C.T., Wang, C.Y., Chien, S.Y., Lai, S.H.: Fedfr: Joint optimization federated framework for generic and personalized face recognition. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. vol. 36, pp. 1656–1664 (2022)
- [12] Liu, Z., Chen, Y., Zhao, Y., Yu, H., Liu, Y., Bao, R., Jiang, J., Nie, Z., Xu, Q., Yang, Q.: Contribution-aware federated learning for smart healthcare. In: *Proceedings of the 34th Annual Conference on Innovative Applications of Artificial Intelligence* (2022)
- [13] Long, G., Tan, Y., Jiang, J., Zhang, C.: Federated learning for open banking. In: *Federated learning*, pp. 240–254. Springer (2020)
- [14] Long, G., Xie, M., Shen, T., Zhou, T., Wang, X., Jiang, J.: Multi-center federated learning: clients clustering for better personalization. *World Wide Web* pp. 1–20 (2022)
- [15] Lou, G., Liu, Y., Zhang, T., Zheng, X.: Stfl: A temporal-spatial federated learning framework for graph neural networks. arXiv preprint arXiv:2111.06750 (2021)

- [16] Ma, Z., Lu, Y., Li, W., Cui, S.: Beyond random selection: A perspective from model inversion in personalized federated learning. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 572–586. Springer (2023)
- [17] Marchand, T., Muzellec, B., Beguier, C., Terrail, J.O.d., Andreux, M.: Securefedyj: a safe feature gaussianization protocol for federated learning. arXiv preprint arXiv:2210.01639 (2022)
- [18] McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)
- [19] Rand, W.M.: Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association* **66**(336), 846–850 (1971)
- [20] Solanki, S., Kanaparth, S., Damle, S., Gujar, S.: Differentially private federated combinatorial bandits with constraints. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 620–637. Springer (2023)
- [21] T Dinh, C., Tran, N., Nguyen, J.: Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems* **33**, 21394–21405 (2020)
- [22] Wang, J., Ma, F.: Federated learning for rare disease detection: a survey. *Rare Disease and Orphan Drugs Journal* (2), 1–14 (2023)
- [23] Wang, J., Qian, C., Cui, S., Glass, L., Ma, F.: Towards federated covid-19 vaccine side effect prediction. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 437–452. Springer (2022)
- [24] Wang, J., Yang, X., Cui, S., Che, L., Lyu, L., Xu, D., Ma, F.: Towards personalized federated learning via heterogeneous model reassembly. *Advances in Neural Information Processing Systems* (2023)
- [25] Wang, J., Zeng, S., Long, Z., Wang, Y., Xiao, H., Ma, F.: Knowledge-enhanced semi-supervised federated learning for aggregating heterogeneous lightweight clients in iot. In: Proceedings of the 2023 SIAM International Conference on Data Mining. pp. 496–504. SIAM (2023)
- [26] Wu, C., Wu, F., Cao, Y., Huang, Y., Xie, X.: Fedgmn: Federated graph neural network for privacy-preserving recommendation. arXiv preprint arXiv:2102.04925 (2021)
- [27] Xie, M., Long, G., Shen, T., Zhou, T., Wang, X., Jiang, J., Zhang, C.: Multi-center federated learning. arXiv preprint arXiv:2005.01026 (2020)
- [28] Zhang, K., Yang, C., Li, X., Sun, L., Yiu, S.M.: Subgraph federated learning with missing neighbor generation. *Advances in Neural Information Processing Systems* **34**, 6671–6682 (2021)
- [29] Zhang, X., Li, Y., Li, W., Guo, K., Shao, Y.: Personalized federated learning via variational bayesian inference. In: International Conference on Machine Learning. pp. 26293–26310. PMLR (2022)