ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230421

Unsupervised Graph Structure-Assisted Personalized Federated Learning

Xiaoying Li[a,b], Xiaojun Chen[b;*], Bisheng Tang[a,b], Shaopu Wang[a,b], Yuexin Xuan[a,b] and Zhendong Zhao[a,b]

[a]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[b]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

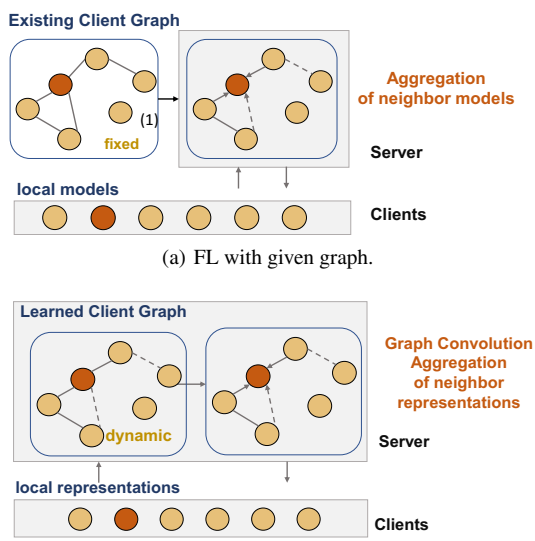
Abstract. Non-IID data presents a significant challenge for federated learning(FL), and personalized FL is a natural solution to address this challenge. Recently, Graph Neural Network (GNN) has recently emerged to model the complex client relationship using a client graph to refine personalized models. However, this approach depends on an existing client relation graph on the server, making it impractical unless this prerequisite is satisfied. Furthermore, noisy and missing connections in the original graph structures can degrade personalization performance. In this work, we propose an unsupervised structure learning approach to improve personalized FL, where the server learns a dynamic client graph through self-supervision and generates structure-based client representations. These representations are then broadcasted to users, regulating local training using the learned knowledge as an inductive bias. Empirical studies on benchmark datasets demonstrate the significant effectiveness of our approach and the high quality of the client graphs. The code is available at https://github.com/lazyJane/FedSKA.

0.1 Introduction

Federated learning (FL) [25] is a rapidly growing collaborative machine learning framework that enables a group of clients to jointly train models without sharing their local data. Classical FL (FedAvg) [41] trains a unique global model for all clients that aims to fit all data from different clients [29, 34, 26, 42]. However, such a global model will always perform poorly due to the statistical heterogeneity observed across different clients [34, 25, 44]. Consider the scenario for mobile device keyboards, certain emojis are used by one demographic but not others. Therefore, it becomes essential to provide client-specific personalized models in FL.

A variety of efforts have been made to personalized FL [6, 21]. One focuses on conducting an additional fine-tuning step after a well-trained global model [49, 39, 1, 9, 40]. However, in highly heterogeneous scenarios, the relevant global model may not exist, and these approaches may result in each client only learning locally. Clustered FL [44, 51, 3, 15, 39] groups clients into several discrete clusters and trains a model for each cluster, where data distribution of clients in one cluster is the same or similar. The Clustered FL assumption is quite restrictive since no knowledge transfer is feasible across clusters. In the scenario where each client has its own unique optimal local model, the number of clusters is equal to the number of clients, rendering FL infeasible. More recently, personalized FL methods [33, 45, 10] are proposed to directly learn many individual personalized models using the global model as the component of knowledge sharing. However, they may not fully exploit the potential pairwise collaborations among clients, as their diversity can provide informative differences in their local data.

Graph learning has emerged as an effective solution to capture relationships among clients, as the connecting edges between clients can depict their correlations [37, 14]. Existing relevant methods mainly focus on network topology applications, such as traffic prediction using sensor networks, and therefore rely on pre-existing client graph. Provided with the client graph as prior knowledge, local models can be improved by capturing inherent information over the topology of FL clients. FedU [11] incorporates a Laplacian regularization term into the objective function to encourage the client models with connected edges to be similar. BiG-Fed [52] utilizes contrastive learning to model the relationship between clients for link prediction on the given client graph. SFL [6] learns a graph-based model for each client by aggregating all models of its neighbors, and then brings the local models closer to the graph-based model.



(a) FL with given graph.

(b) Our FL with learned graph

Figure 1. Concept maps of (a) FL with a given graph and (b) our proposed FL with unsupervised learned graph

However, the prerequisite of the exsiting client graph can leave

* Corresponding Author. Emails:{lixiaoying, chenxiaojun, tangbisheng, wangshaopu, xuanyuexin, zhaozhendong }@iie.ac.cn.

such an approach infeasible for many applications, where a carefully engineered client graph may not always be available on the server. Moreover, the predefined client graph is static, which cannot capture dynamic changes in client relationships that may occur during the FL training process. Even with a known graph, personalization performance may be degraded due to noisy and missing connections in the original graph structures. Moreover, by only consider parameters-level aggregating of neighboring clients, rich and complex information carried in the original data may not be fully utilized.

In this paper, we propose FedSKA, a personalized FL method that leverages dynamic structured knowledge. FedSKA utilizes a structure learner to generate a dynamic client graph during FL training through self-supervision based on homophily of the graph. The learned weighted adjacency matrix is then fed into a Graph Neural Network (GNN) model to produce representations with structural knowledge, which is optimized by borrowing gradient information from clients, thereby leveraging the inductive bias from local tasks. These structure-based features are then broadcasted to clients, enabling them to train their models on the learned graph structure that embodies knowledge from neighboring clients.

Our main contributions are as follows :

1. Problem. We propose to regulate local model updating by imposing structure relational inductive biases, without relying on any external prior knowledge about clients, which leads to better personalized performance under non-i.i.d. data distributions. To our best knowledge, we are the first attempt to explore client structure through unsupervision under FL, which is more practical to address more challenging FL scenarios.
2. Algorithm: We propose the dynamic Structural Knowledge Assisted framework FedSKA, which learns a client graph by finding the optimal structure that facilitates to predict the client representations. FedSKA performs an alternating and periodical process to implement structure knowledge transfer between clients and server, incorporating benefits from both FedAvg and Split Learning.
3. Evaluations. We perform extensive experiments to corroborate the effectiveness and analyze the properties of FedSKA through comparisons with state-of-the-art methods on four benchmark datasets.

1 Related Work

1.1 Federated Learning for Non-IID Data

The vanilla FL algorithm FedAvg [41] trains a unique model to fit data across various clients. However, it suffers from the non-i.i.d. decentralized data, leading to statistical challenges such as model weights divergence [54], data distribution biases [20], and a drifted global model that is slow to converge even unguaranteed convergence [34]. To tackle this challenge, Li *et al.* [34] proposed FedProx, which adds a proximal term to the local objective function to reduce the gap between local and global models. SCAFFOLD [26] introduces control variates to correct the client drift in its local updates. 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.

1.2 Personalized Federated Learning

Given the challenges described above, some other approaches adopt the strategy of training multiple models or personalizing components to address multiple target distributions.

Cluster-level PFL Clustered FL assumes that the clients can be partitioned into different clusters, representing different distributions. CFL [44] recursively separate clients with incongruent optimization directions by the cosine similarity of the parameter updates, while FedSEM [51] uses l_2 distance. Briggs *et al.* [3] propose an agglomerative hierarchical clustering method named FL+HC, which relies on iterative calculating the pairwise distance between all clusters. IFCA [15, 39] divides the clients into clusters with a center model that can minimize their loss values.

Client-level PFL typically assumes that each client's data distribution is unique, which necessitates the training of a personalized model on each client's device. A natural approach is learning a global model and fine-tuning parameters on each client's local dataset [49, 39, 1, 9, 40]. Per-FedAvg [12] and a class of algorithms referred to as meta-learning [24, 5, 27] considered fine-tune as a regularization term on the learning objective function of the global model. Ditto [33] proposed a bilevel optimization framework for PFL, which incorporates a regularization term to constrain the distance between the local and global model. Interpolation of global and local model [39, 19, 31] build personalized models for clients by combining the global model and the local model. These approaches primarily focus on the interaction between the local and global models, thereby failing to discover differences among the diverse information across clients.

Graph-assisted PFL Graph structure has recently emerged as an effective approach to model the relationships among the clients. [37, 14]. FedU [11] reformulates a multi-task federated learning using Laplacian regularization. BiG-Fed [52] designs a bilevel optimization framework and leverages the connectivity of edges as a guiding information in the outer level task by mapping into the structural similarity of neighboring node models. SFL [6] learns graph-based personalized models and then brings the local models closer to the graph-based model. The quality of the client graph and the effective incorporation of structural information for each client are two critical factors in Graph-assisted FL. Our proposed method addresses both of these issues to improve performance.

1.3 Graph Neural Networks

Graph Neural Networks (GNNs) have demonstrated superior performance in various learning tasks involving graph-structured data, such as graph embedding [18] and node classification [28]. GNNs capture the relationships between nodes in a graph using k-hop aggregation, and a weighted hop in GCN can capture more complex relationships in the graph [7]. GCN, proposed by Kipf and Welling [28], performs convolutional operations on graph-structured data. In our method, GCN is used to fuse neighbor clients' information.

1.4 Graph Structure Learning

Graph Structure Learning (GSL) improves the robustness of GNN models by jointly learning an optimized graph structure and corresponding representations [55, 56]. Some methods attempts to reweight the existing edges of the given graph using attention mechanism or feature similarity [46, 22]. Other methods reconstruct structures with node-wise similarity computed by metric learning functions like cosine similarity [8] and dot production [53]. Then the adjacency matrix with learnable parameters via jointly optimized along with GNN under the supervision task such as node classification [8]. SLAPS [13] identified a supervision starvation problem that

emerges for graph structure learning and perform GSL through unsupervision. SUBLIME [38] guides structure optimization by maximizing the agreement between the learned structure and a crafted self-enhanced learning target with contrastive learning.

2 Preliminary and Notation

In this paper, we consider a general supervised learning task in the entire dataset $\mathcal{D}$ under federated setting. Given K clients in an FL system, the k th client has its own dataset $\mathcal{D}_k := \left\{\left(\boldsymbol{X}_k^i, y_k^i\right)\right\}_{i=1}^{N_k}$, where $\boldsymbol{X}_k^i$ is the i th training sample, y_k^i is the corresponding ground truth of $\boldsymbol{X}_k^i$. N_k is the sample number in dataset $\mathcal{D}_k$ and the total number of data $N = \sum_{k=1}^{K} N_k$. In general, the goal of vanilla FL system is to solve the following objective function:

$$\min_{w} F(f_1(w), ..., f_K(w)) \tag{1}$$

where $f_k(w) := \mathbb{E}_{\boldsymbol{X}_k \sim \mathcal{D}_k}\left[l_k\left(w; \boldsymbol{X}_k\right)\right]$, is the k th client's local objective function that measures the local empirical risk over $\mathcal{D}_k$. $F(\cdot)$ is a function that aggregates the local objectives from each client. For example, in FedAvg [41], $F(\cdot)$ is typically set to be a weighted average of local losses, i.e., $\sum_{k=1}^{K} p_k f_k(w)$, where $p_k = \frac{N_k}{N}$ and $\sum_k p_k = 1$. However, The heterogeneity of $\mathcal{D}$ can often prevent convergence to a stable global solution for Eq. (1). To address this issue, it is common to learn client-specific models $\{w_k\}_{k\in[K]}$ for personalization.

The formulation of a personalized FL system typically involves a bi-level optimization problem as follows:

$$\begin{array}{ll} \min_{w_1,\ldots,w_k} & h_i\left(w_k; w^*\right) := f_i\left(w_k\right) + \lambda R\left(w_k, w^*\right) \\ \text{s.t.} & w^* \in \arg\min_w F\left(f_1(w), \ldots, f_N(w)\right) \end{array} \tag{2}$$

where each client has a personalized model w_k, and w^* is the optimal global model that minimizes the loss in Eq. (1). The regularization term R controls the local model updates. Ditto [33] propose an l_2 term $\frac{1}{2}\left|w_k - w^*\right|^2$ to constrain the local updating to be close to the global model. As a typical method of graph-assisted FL, SFL first learns a graph-based aggregation model u and then introduces an l_2 term $\frac{1}{2}\left|w_k - u\right|^2$ to constrain the local updates to be close to the graph-based model. Our approach differs from theirs in that we utilize feature representations to convey structural information for sufficient representation learning capabilities.

3 Proposed method: FedSKA

In this section, we elaborate our proposed approach with a summary shown in Algorithm 1. An overview of its learning procedure in illustrated in Figure 2.

3.1 Formulation

Before making the problem statement of FedSKA, we first introduce the basic definition of the client graph in FL. Consider a connected graph $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{H}^c) = (\boldsymbol{S}, \boldsymbol{H}^c)$, where $\mathcal{V}$ is the set of $K = |\mathcal{V}|$ nodes, each node represents a client, $\mathcal{E}$ is the set of $m = |\mathcal{E}|$ edges representing connections between clients. $\boldsymbol{S} \in [0,1]^{K\times K}$ is the weighted adjacency matrix. $\boldsymbol{H}^c = \{\boldsymbol{h}_k^c\}_{k=1}^K$ is the node feature matrix, where each row $\mathbf{h}_k^c$ denotes the feature representation of client k produced by the local feature extractor f_k^e with parameters $\boldsymbol{w}_k^e$ on the local data $\boldsymbol{X}_k$ as follows: $\boldsymbol{h}_k^c = f_k^e(\boldsymbol{w}_k^e; \boldsymbol{X}_k)$.

The design of FedSKA is motivated by the lack and poor quality of the original static graph that may not capture dynamically changing relationships between clients during training. Hence, our goal is to capture the latent structure among clients and use it to generate robust representations of neighboring clients to improve the personalized accuracy performance. Our Structure Knowledge-Assisted personalized FL Framework considers bi-level tasks.

$$\begin{aligned} &\textit{Server}: \boldsymbol{S} = G_S(\boldsymbol{H}^c), \quad \boldsymbol{H}^g \leftarrow \mathrm{GNN}_E(\boldsymbol{S}, \boldsymbol{H}^c) \\ &\textit{Client}: \min_{w_1,\ldots,w_k} f_k(w_k) := \mathcal{L}_C(w_k) + \mathcal{L}_S(\boldsymbol{H}^c, \boldsymbol{H}^g) \end{aligned} \tag{3}$$

where the server's GNN task involves two key components: a structure learner G_S responsible for constructing the client graph $\boldsymbol{S}$, and a GNN encoder GNN_E tasked with updating the hidden features $\boldsymbol{H}^c$ to more intricate structure-based features $\boldsymbol{H}^g$. The clients are involved in a local task that aims to minimize the local empirical risk $\mathcal{L}_C(w_k)$ while leveraging the structure-based features $\boldsymbol{H}^g$ obtained from the client graph $\boldsymbol{S}$ learned by the server. We show later that the learned structure-based features can introduce inductive bias to local users, reinforcing their model learning with a better generalization performance. In the following sections, we will provide a detailed description of each module of our approach.

3.2 Client Structure Learner

The structure learner G_S is a function $G_S : \mathbb{R}^{K\times d} \to \mathbb{R}^{K\times K}$ with parameters θ_{G_S} which takes the client features $\boldsymbol{H}^c \in \mathbb{R}^{K\times d}$ as input and generates a adjacency matrix $\tilde{\boldsymbol{S}} \in \mathbb{R}^{K\times K}$ as output, d is the dimension of client features. The process of client graph structure learning follows the general graph structure learning paradigm: 1) generator, 2) adjacency processor, and 3) learning objectives. Below we provide details for each module.

3.2.1 Metric-based Generator

To find optimal structures for various information across clients, we design a metric learning based generator G_S, which first acquires node embeddings from the input data using a multi-layer perceptron (MLP), and then model structure with pair-wise similarity of the node embeddings. Specially, $\tilde{\boldsymbol{S}}$ is generated by

$$\tilde{\boldsymbol{S}} = G_S(\boldsymbol{H}_e^c) = \sigma(G_{MLP}(\boldsymbol{H}^c) G_{MLP}(\boldsymbol{H}^c)^{\mathsf{T}}) \tag{4}$$

where G_{MLP} is a multi-layer perceptron (MLP) with the learnable parameters $\theta_{G_{MLP}}$ to generate the embedding representations $\boldsymbol{H}_e^c$, which considers the correlation and combination of features, generating more informative embeddings for downstream similarity metric learning. $\sigma(\cdot)$ is a non-linear activation and defined as ReLU function, which make sure all elements to be positive.

3.2.2 Post-processor

The output $\tilde{\boldsymbol{S}}$ of the generator may be dense, non-symmetric and non-normalized, which violates the properties of real-world graphs. Thus, we apply the post-processor $T(\cdot)$ to refine the sketched adjacency matrix $\tilde{\boldsymbol{S}}$ into a sparse, symmetric, and normalized adjacency matrix $\boldsymbol{S}$. This is achieved through three post-processing steps applied sequentially: sparsification, symmetrization, and normalization.

Sparsification: The dense sketched adjacency matrix $\tilde{\boldsymbol{S}}$ often contains noise (i.e. unimportant edges) and increases the computational burden, especially when the FL system has a large number of

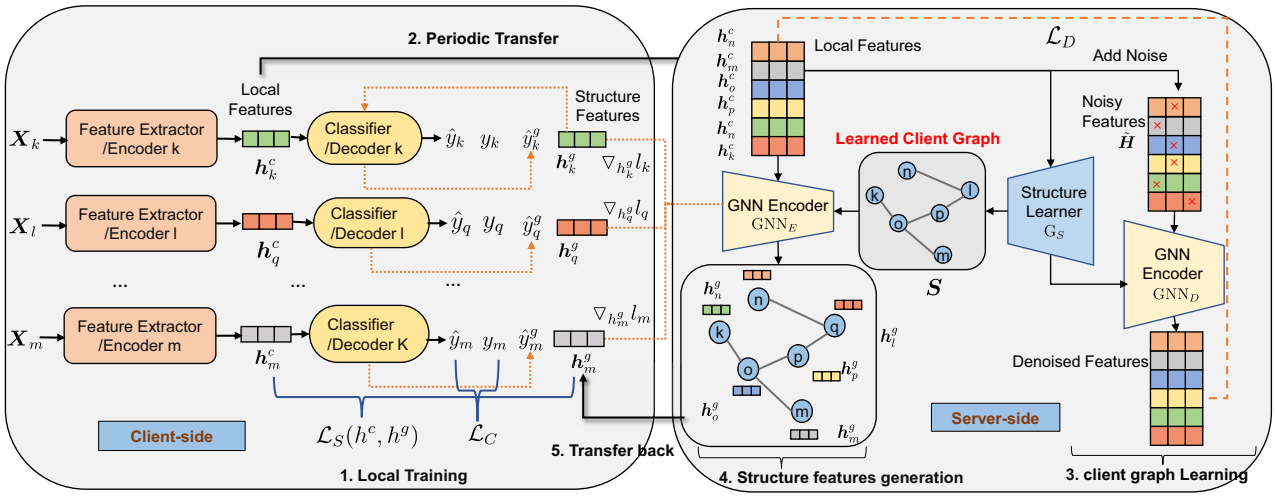


Figure 2. Framework of FedSKA

Algorithm 1 FedSKA. E_c is the number of local epochs, E_S is the number of structure learning rounds, E_g is the number of GNN_E learning rounds, T is the number of communication rounds; η is the learning rate. $\boldsymbol{h}_k^c$ is the local features of client k, $\boldsymbol{h}_k^g$ is the structure-based features generated by server for client k.

ServerExecute()
for each round $t = 0, ..., T$ **do**
  //(1)**Optional FedAvg of local models.**
  for each client $k \in \mathcal{V}$ **in parallel do**
    $\boldsymbol{h}_k^c \leftarrow$ **ClientUpdate**$(k, \boldsymbol{h}_k^g)$
  end for
  $w_t = \sum \frac{N_k}{N} w_k$
  for each client $k \in \mathcal{V}$ **in parallel do**
    Send w_t to reinitialize client w_k
  end for
  $\boldsymbol{H}_k^c[idx] \leftarrow \boldsymbol{h}_k^c$
  //(2)**Unsupervised Client Structure Learning.**
  for server round from 1 to E_S **do**
    $\tilde{\boldsymbol{S}} \leftarrow$ Metric-based Generator $\mathrm{G}_{MLP}(\boldsymbol{H}_c)$
    $\boldsymbol{S} \leftarrow$ Post-processor $T(\tilde{\boldsymbol{S}})$
  end for
  //(3)**Alternate optimization of** GNN_E
  for each round from 1 to E_g **do**
    $\{\boldsymbol{h}_k^g | i \in \mathcal{V}\} \leftarrow \mathrm{GNN}_E(\{\boldsymbol{h}_k^c | i \in \mathcal{V}\}; \theta_{\mathrm{GNN}_E})$
    for each client $k \in \mathcal{V}$ **in parallel do**
      $\nabla_{\boldsymbol{h}_k^g} l_k \leftarrow l_k \cdot Backward(k, \boldsymbol{h}_k^g)$
      $\nabla_{\theta_{\mathrm{GNN}_E}} l_k \leftarrow \boldsymbol{h}_k^g \cdot Backward(\nabla_{H_g} l_k)$
    end for
    $\nabla_{\theta_{\mathrm{GNN}_E}} l \leftarrow \sum_{k \in \mathcal{V}} \nabla_{\theta_{\mathrm{GNN}_E}} l_k$
    $\theta_{\mathrm{GNN}_E} \leftarrow \theta_{\mathrm{GNN}_E} - \eta \nabla_{\theta_{\mathrm{GNN}_E}} l$
  end for
  //(4)**Update structure-based features**
  $\boldsymbol{H}_g \leftarrow \mathrm{GNN}_E(S, \boldsymbol{H}_c)$
  for each client $k \in \mathcal{V}$ **in parallel do**
    send $\boldsymbol{h}_k^g$ to client k
  end for
end for
ClientUpdate$(k, \boldsymbol{h}_k^g)$:
//(1)**Local Training**
for each local epoch from 1 to E_c **do**
  for batch $b \in \{X_k, Y_k, \boldsymbol{h}_k^g\}$ **do**
    Update w_k as in Eq. (9)
  end for
  //(2)**Extract Local features**
  for idx, batch $x_k, y_k \in \{X_k, Y_k\}$ **do**
    $\boldsymbol{h}_k^c = f_k^e(\boldsymbol{w}_k^e; \boldsymbol{x}_k^c)$
  end for
  return $\boldsymbol{h}_k^c$ to server
end for
ClientBackward$(k, \boldsymbol{h}_k^g)$:
$\hat{y}_k \leftarrow f_k^s(w_k^l, \boldsymbol{h}_k^g)$
$l_k \leftarrow l_k(\hat{y}_k, y_k)$
return $\nabla_{\boldsymbol{h}_k^g} l_k$ to server

clients [50]. So we consider only the "K-nearest neighbor" for each client to produce a sparse matrix. Specifically, we keep the edges with top-k connection values and mask off the rest (set to 0). The sparsification $(\cdot)$ is expressed as:

$$\tilde{\boldsymbol{S}}_{ij} = T_{sp}\left(\tilde{\boldsymbol{S}}_{ij}\right) = \begin{cases} \tilde{\boldsymbol{S}}_{ij}, & \tilde{\boldsymbol{S}}_{ij} \in \text{top}-\text{k}\left(\tilde{\boldsymbol{S}}_i\right) \\ 0, & \tilde{\boldsymbol{S}}_{ij} \notin \text{top}-\text{k}\left(\tilde{\boldsymbol{S}}_i\right) \end{cases} \tag{5}$$

Symmetrization and Normalization. The process can be achieved as follows:

$$\begin{aligned} \boldsymbol{S} &= T_{\text{norm}}\left(T_{\text{sym}}\left(\tilde{\boldsymbol{S}}\right)\right) \\ &= T_{\text{norm}}\left(\frac{1}{2}(\tilde{\boldsymbol{S}} + \tilde{\boldsymbol{S}})^\top\right) \\ &= \frac{1}{2}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\left(\tilde{\boldsymbol{S}} + \tilde{\boldsymbol{S}}^\top\right)\tilde{\boldsymbol{D}}^{-\frac{1}{2}} \end{aligned} \tag{6}$$

where T_{sym} makes the adjacency matrix symmetric, which takes the average of the similarities to ensure that the strength of the connec-

tion between v_i and v_j is the same in both directions. In general, if client A is more important to client B, then client B is also more important to client A. Therefore, to reflect such an undirected graph, the adjacency matrix needs to be symmetrized. Finally, to normalize the symmetric adjacency matrix $\tilde{\boldsymbol{S}}$, we compute its degree matrix $\tilde{\boldsymbol{D}}$ and then multiply $\tilde{\boldsymbol{D}}^{-\frac{1}{2}}$ from the left and right to $\tilde{\boldsymbol{S}}$ to ensure that the sum of rows and the sum of columns are equal to 1.

3.2.3 Optimization by Self-supervision

Since the server has no access to client data, there is an absence of prior knowledge about the clients. Therefore, we optimize the parameterized adjacency matrix $\boldsymbol{S}$ through unsupervised learning. Inspired by contrastive learning [48], we design a supervision signal from data itself via augmented representations. Specifically, we create augmented features by adding noise to the original client features. Then, we feed the learned adjacency matrix $\boldsymbol{S}$ and the augmented features into a Graph Neural Network (GNN) to generate denoising features. Finally, we maximize the similarity between the denoising features and the original features. Our self-supervised task encourages the model to learn a graph structure suitable for predicting the node features, satisfying the homophily of the graph [28]. $\mathrm{GNN}_D : \mathbb{R}^{K\times d} \times \mathbb{R}^{K\times K} \to \mathbb{R}^{K\times d}$ is a GNN with learnable parameters θ_{GNN_D}, which takes noisy version $\tilde{\boldsymbol{H}}$ of client features as inputs and produces denoised features $\boldsymbol{H}$ with the same dimension as output. During training, we minimize:

$$\mathcal{L}_D = \mathrm{L}\left(\boldsymbol{H}_{pos}, \mathrm{GNN}_{\mathrm{D}}\left(\tilde{\boldsymbol{H}}, \boldsymbol{S}; \theta_{\mathrm{GNN}_{\mathrm{D}}}\right)_{pos}\right)$$

where pos represents the indices corresponding to the elements of $\boldsymbol{H}$ to which we have added noise, and $\boldsymbol{H}_{pos}$ represents the values at these indices. $\boldsymbol{S}$ is the generated adjacency matrix learned by G_S and L is the mean-squared error loss. pos consists of r percent of the indices of $\boldsymbol{H}$ selected uniformly at random in each epoch. Adding noise can be performed by either replacing the values at pos with 0 or by adding independent Gaussian noises to each of the features.

3.3 Structure-based Representations Learning

After obtaining the learned client graph, we use a GNN to update the client feature representations to fuse information about the neighboring clients. Specifically, we employ Eq. (7) to update the client feature embeddings:

$$\{\boldsymbol{h}_k^g\}_{k=1}^K \leftarrow \mathrm{GNN}_E(\boldsymbol{S}, \{\boldsymbol{h}_k^c\}_{k=1}^K; \theta_{\mathrm{GNN}_E}) \tag{7}$$

where θ_{GNN_E} are the trainable model parameters. By iteratively update the feature embeddings, GNN_E captures the complex information of the client representations over the topology.

To optimize θ_{GNN_E}, we adopt gradient information borrowed from the clients using Split Learning (SL) [17, 47]. First, the server obtains the updated hidden feature $\boldsymbol{H}^g$ through GNN_E and sends the k-th row feature $\boldsymbol{h}_k^g$ to the corresponding client k. Then, the client k feeds $\boldsymbol{h}_k^g$ to its local classifier f_k^s to compute the loss with respect to $\boldsymbol{h}_k^g$. Next, the client computes the gradient $\nabla_{\boldsymbol{h}^g} l_k$ using local gradient back propagation as in Eq. (8a), and sends it back to the server. The server receives the gradients $\{\nabla_{\boldsymbol{h}^g} l_k\}_{k=1}^K$ from all clients and computes each gradient of l_k with respect to GNN_E, denoted as $\nabla_{\theta_{\mathrm{GNN}_E}} l_k$, using Eq. (8b). Finally, the server aggregates all gradients $\{\nabla_{\theta_{\mathrm{GNN}_E}} l_k\}_{k=1}^K$ and obtains the result gradient using Eq. (8c). The parameters θ_{GNN_E} are updated using gradient descent as in Eq. (8d), where η is the learning rate.

$$\textit{Client } k: \nabla_{\boldsymbol{h}_k^g} l_k \leftarrow l_k.Backward(k, \boldsymbol{h}_k^g) \tag{8a}$$

$$\textit{Server}: \nabla_{\theta_{\mathrm{GNN}_E}} l_k \leftarrow \boldsymbol{h}_k^g.Backward(\nabla_{\boldsymbol{h}_k^g} l_k) \tag{8b}$$

$$\nabla_{\theta_{\mathrm{GNN}_E}} l \leftarrow \sum_{k\in\mathcal{V}} \nabla_{\theta_{\mathrm{GNN}_E}} l_k \tag{8c}$$

$$\theta_{\mathrm{GNN}_E} \leftarrow \theta_{\mathrm{GNN}_E} - \eta \nabla_{\theta_{\mathrm{GNN}_E}} l \tag{8d}$$

To alleviate the high communication cost for exchanging information, we use alternate optimization algorithm [2] for periodic information exchange as in Figure2.

By this approach, FedSKA takes into account the task-specific inductive bias of local data during the learning process, thereby improving the GNN_E parameters and making them more suitable for downstream local tasks. At the end of learning process, the learned feature $\boldsymbol{h}_k^g$ is sent to the corresponding client. As a result, the objective of a local model k is altered to maximize the similarity between local features $\boldsymbol{h}_k^c$ and structure-based features $\boldsymbol{h}_k^g$:

$$\min f_k(w_k) := \mathcal{L}_C(w_k) + \mathcal{L}_S(\boldsymbol{h}_k^c, \boldsymbol{h}_k^g) \tag{9}$$

where $\mathcal{L}_C(w_k)$ is the empirical risk of client k, and $\mathcal{L}_S(\boldsymbol{h}_k^c, \boldsymbol{h}_k^g)$ is the similarity-based loss that brings the local features closer to the received structure-based features, as in Eq. (10):

$$\mathcal{L}_S(\boldsymbol{h}_k^c, \boldsymbol{h}_k^g) = \frac{1}{N_k}\sum_{i=1}^{N_k} 1 - cos(h_{c,k}^i, h_{g,k}^i) \tag{10}$$

It is worth noting that the structure-based features $\boldsymbol{h}_k^g$ combine a client's local features with those of neighbors through weighted sums based on the edge weights of the client graph. The highest weight is assigned to the self for its highest self-similarity. This ensures the incorporation of useful neighbor information without neglecting local information, resulting in stable personalized model performance improvement. Moreover, compared to traditional FL, FedSKA enhances collaboration among neighboring clients through structure learning optimization. This involves updating variables $\boldsymbol{S}$ to find the most similar clients and w_k for local model updates from local data, resulting in mutually reinforcing convergence. Convergence analysis will be addressed in our future work.

4 Experiments

4.1 Datasets and models

We evaluated our method on four federated benchmark datasets spanning a wide range of machine learning tasks: MNIST [32] for handwritten character recognition, CIFAR10 [30] for image classification, Shakespeare [4, 41] for language modeling, and METR-LA [36] for traffic forecasting. Client-side models used include: a 1-hidden layer MCLR model with 64 units for MNIST, MobileNet-v2 [43] with a last hidden size of 1280 for CIFAR10, a stacked RNN with an embedding layer and a 1-layer LSTM [16] with hidden size 256 and a linear layer for Shakespeare, and a 2-layer GRU model with hidden size 64 for METR-LA. On the server side, we employed an MLP with the same hidden size as the local feature and ReLU activation for the structure learner, and a 1-layer Graph Network as the GNN Encoder. All datasets were randomly split into 80% training and 20% test sets for each task.

Table 1. Personalized Performance comparisons on different settings

Dataset	Setting		Separate	FedAvg	FedProx	Ditto	IFCA	FedU	SFL	Ours
MNIST	shards=2	Mean	98.34±0.01	90.02±0.00	90.01±0.02	98.34±0.01	97.32±0.02	98.28±0.00	97.17±0.01	**98.65**±0.01
		Best	100.00±0.00	98.00±0.00	97.82±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	**100.00±0.00**
		Worst	**94.65±0.31**	81.08±0.00	81.08±0.00	94.65 ±0.31	92.65±0.00	94.74±0.31	91.78±0.00	93.19±0.31
	shards=5	Mean	93.87±0.00	90.20±0.02	90.06±0.01	93.91±0.00	93.38±0.01	**94.56±0.00**	93.11±0.01	94.37±0.00
		Best	99.54±0.00	96.77±0.00	96.77±0.00	98.14 ±0.00	99.14 ±0.00	99.54±0.00	99.23±0.00	**99.84±0.00**
		Worst	87.60±0.38	83.69±0.00	83.69±0.00	87.14±0.00	87.57±0.00	88.4±0.00	87.08±0.00	**88.92±0.00**
	shards=10	Mean	90.17±0.01	90.53±0.01	90.42±0.01	90.17±0.01	91.73±0.00	92.18±0.02	91.48±0.01	**92.42±0.00**
		Best	96.77±0.00	96.46±0.00	96.31±0.00	96.77±0.00	98.15±0.00	96.77±0.00	97.75±0.00	**98.59±0.00**
		Worst	84.12±0.00	84.31±0.00	84.15±0.00	84.12±0.00	86.62±0.00	87.38±0.00	85.69±0.00	**87.72±0.00**
CIFAR10	shards=2	Mean	82.66±0.05	69.91±0.11	58.84±0.26	88.34±0.01	93.54±0.09	84.96±0.09	93.99±0.07	**94.87±0.02**
		Best	100.00±0.00	96.70±0.33	94.53±0.00	96.70±0.33	100.00±0.00	100.00±0.00	100.00±0.00	**100.00±0.00**
		Worst	51.63±0.62	28.57±1.58	2.50±0.00	68.93±1.00	80.77±0.41	57.27±1.22	79.50±0.67	**78.54±0.93**
	shards=5	Mean	41.1±0.12	71.85±0.03	62.83±0.26	41.36±0.23	85.21±0.10	40.55±2.48	87.85±0.06	**91.27±0.16**
		Best	79.53±0.00	84.77±0.53	80.40±0.63	81.75±0.00	95.97±0.00	79.36±0.98	96.87±0.34	**98.04±0.51**
		Worst	21.34±0.41	55.63±0.34	44.97±0.86	22.02±0.00	70.39±0.53	21.58±0.86	75.46±0.63	**80.21±0.66**
	shards=10	Mean	37.47±0.06	75.45±0.04	73.41±0.02	30.69±0.11	81.32±0.12	47.33±0.09	85.44±0.07	**90.1±0.06**
		Best	87.93±1.02	87.93±1.02	86.64±0.34	60.64±0.63	93.53±0.68	60.03±0.42	94.37±0.42	**98.04±0.00**
		Worst	18.75±0.34	64.88±0.34	61.90±0.68	17.49±0.63	66.47±0.63	29.69±0.68	77.49±0.42	**82.88±0.51**
Shakespeare		Mean	32.01±0.04	**31.18±0.01**	21.71±0.06	29.46±0.00	27.87±0.02	30.01±0.03	30.18±0.07	30.41±0.22
		Best	29.22±0.04	33.56±0.02	25.13±0.85	22.82±0.00	30.86±0.02	26.76±0.02	32.02±0.22	**34.19±0.62**
		Worst	22.83±0.01	27.12±0.01	20.45±0.17	15.04±0.02	25.12±0.21	18.39±0.07	27.10±0.15	**31.51±0.14**

Client heteroeneity Setting. Shakespeare dataset is naturally non-i.i.d distributed where each client represents a character. For MNIST and CIFAR-10, we artificially partitioned the raw dataset using a parameter q (shards) to control the level of heterogeneity, following the approach of [41]. We sorted the entire dataset according to the label and divided it into $K \times q$ shards of equal size, with each client assigned q shards. The level of non-i.i.d. data issues depends on the size of the shards, with smaller shards indicating higher data heterogeneity. METR-LA [36] is a traffic dataset that has a graph topology connecting sensors on roads. Each sensor on the road can be considered a client in the FL system, contributing data collected from real-world sources with a non-i.i.d. distribution.

4.2 Baseline and experiment settings

We compared our algorithms with one global model trained with **FedAvg** [41] and **FedProx** [35] . Personalized FL **Ditto** [33] sets up a separate personalization model and make it close to the global model. **IFCA** [15, 39] groups similar clients in to the same cluster and performs FedAvg in each cluser independently. **FedU** [11] aggregates neighbor models for each client. **SFL** [6] learns graph-based personalized models with an optional l_2 distance-based graph construction method.

Without any additional statement, all reported results are based on the same training configuration. We employ SGD with the same learning rate 0.01 as the optimizer for all training operations, use a batch size of 32. For the training process, we choose the number of local epochs $E_c = 1$, the number of structure learning rounds $E_S = 10$ and the number of GNN_E learning rounds $E_g = 10$ for all datasets. The number of total communication T is rounded to 200. The feature mask ratio $r = 0.01$.

4.3 Performance Comparison analysis

Table 1 reports the average, the best 5% and the worst 5% performance of personalized models of our method and other baselines on MNIST, CIFAR10 and Shakespeare, respectively. Table 2 reports the average MAE, MAPE, and RMSE across all clients for predicting 60 minutes (12-time steps) ahead on the METR-LA dataset. And we have the following findings from the results.

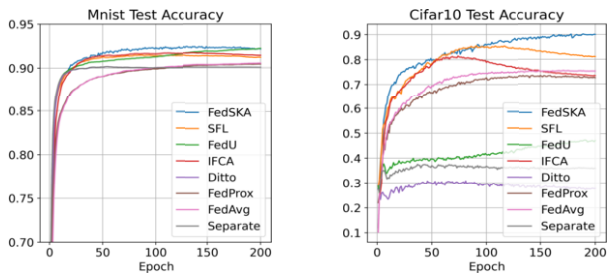
- **Overall Performance.**
  - *Performance on MCLR and CNN*. FedSKA significantly improves test accuracy on image datasets, achieving approximately 8% improvement on MNIST and 27% improvement on CIFAR10. This demonstrates its effectiveness, particularly for complex local classification tasks. In comparison, FedProx and Ditto rely on average models' and local models' performance, respectively, since their simple regularization approaches may not adequately address underlying heterogeneity, leading to unstable performance. IFCA alleviates heterogeneity to a certain extent, but its effectiveness is limited by the absence of knowledge interaction between clusters. FedU benefits from a given client graph on MNIST but can easily fail under complex tasks like CIFAR10, where a more sophisticated client structure is required. As one of the most competitive baselines, SFL improves personalized performance compared to FedU by constructing a client graph based on l_2 distance between local model parameters. However, its gain is less significant compared to FedSKA. We attribute the superior performance of FedSKA to the learned robust latent structure knowledge by self-supervision, which effectively enhances personalized models.
  - *Performance on RNN*. FedSKA outperforms other baselines on RNN, as shown in the last part of Table 1 and Table 2. For RNN with a large number of parameters and complex network structure, FedAvg achieves stable average performance by aggregating all clients' information. However, the best 5% and worst 5% personalized performances show that there is significant variance between client models, which violates fairness in FL. In contrast, FedSKA maintains relatively stable performance for all client models by finding optimal neighbors and appropriate edge weights for information transmission.
- **Impacts of data heterogeneity.** FedSKA is robust against different levels of user heterogeneity and consistently performs well. As the number of shards becomes smaller (higher heterogeneity), traditional FL algorithms like FedAvg and FedProx lose effectiveness and may perform worse than Separate training due to severe heterogeneity affecting the global model performance. However, FedSKA remains effective and accurately captures client structural relationships, accommodating clients with varying levels of

Table 2. Performance of traffic forecasting in FL

METR-LA	MAE	MAPE	RMSE
Separate	0.3680	0.4492	4.2644
FedAvg	0.3497	0.9352	3.5631
FedProx	0.3539	0.4043	3.6229
Ditto	0.3669	0.3952	4.3251
IFCA	0.3498	0.4300	**3.5270**
FedU	0.3659	0.4440	4.1447
SFL	0.3656	0.4001	3.6406
FedSKA	**0.3465**	**0.3853**	3.5348

closeness.

- **Learning efficiency.** As depicted in Figure 3, FedSKA exhibits the most rapid learning curves, reaching optimal performance quickly. Although FedSFL demonstrates higher learning efficiency than other baselines, it is susceptible to overfitting since the fixed graph structure leads a client to always exchange information with its fixed and limited neighbor nodes, which is also observed with IFCA. In contrast, by dynamically adapting the learned struture knowledge to clients, FedSKA can customize the training process for each client and achieve better overall performance.



(a) MNIST-shards10 (b) CIFAR10-shards10

Figure 3. Visualization of Convergence

4.4 *Ablation Study*

In this section, we analyze the effectiveness of the learned structure. For comparison, we created simple initial graphs for image datasets by connecting two clients with the same label. We then compared our method using the pre-defined graph and the learned graph by the client structure learner. Results in Table 3 show that the learned client structure is crucial for exploring latent relationships between clients, which are otherwise inaccessible with the given graph.

Table 3. Personalized performance of ablation experiments

Average Test Acc	MNIST	CIFAR10
FedSKA (given graph)	93.90	89.60
FedSKA (learned graph)	**94.60**	**91.47**

4.5 *Number of neighbors k*

We investigate the sensitivity of the number of neighbors k in kNN for sparsification. We search the range 5, 10, ..., 80 for MNIST and CIFAR10. Figure 4 demonstrates that FedSKA is robust under different values of k. The optimal selection for each dataset is $k = 70$ for MNIST and $k = 20$ for CIFAR10. This is because the local data for MNIST is relatively simple, necessitating a larger number of neighbors to provide sufficient information, whereas CIFAR10 requires identifying accurate neighbor aggregation for achieving optimal performance.

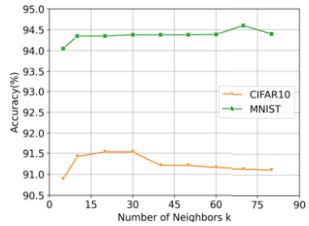
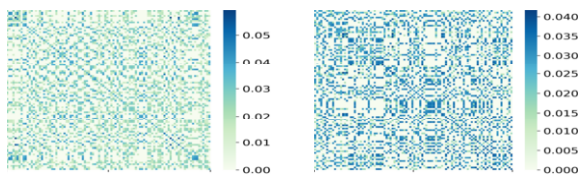


Figure 4. Average personalized test accuracy w.r.t. number of neighbors.

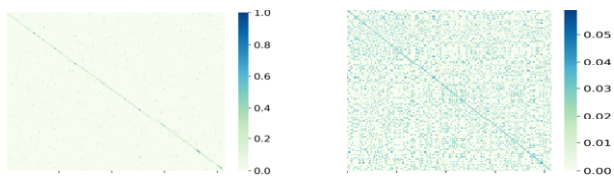
4.6 *Visualization of learned client structure*

Figure 5 visualize the learned graph with $k = 30$ on MNIST for shards2 and shards10. It can be observed that when the heterogeneity is high (shards=2), the similarity between clients is small, resulting in lower connection weights learned for the graph edges. Figure 6 visualizes the comparison between the pre-defined relation graph and the learned graph by FedSKA on METR-LA. The learned graph on METR-LA finds more hidden connection edges, including the long-dependence on non-connected roads. This visualization demonstrates that FedSKA not only learns knowledge from simple pre-defined graphs but also discovers complex hidden relationships among clients.



(a) Learned Graph for shards2 (b) Learned Graph for shards10

Figure 5. The visualization of the learned client graph $\boldsymbol{S}$ for MNIST vs. different heterogeneity.



(a) Original Graph (b) Learned Graph by FedSKA

Figure 6. The visualization of the learned client graph $\boldsymbol{S}$ for METR-LA.

5 Conclusion

In this work, we propose a novel approach to improve personalized performance using unsupervised graph structure learning. We accomplish this through a bi-level task in FL, where on the server side, a dynamic client graph is learned through self-supervision and representation is generated by a GN that leverages local downstream task. On the client side, the local objective is designed to be close to the learned representation which incorporates the learned structure. We conduct experiments to evaluate the effectiveness of the proposed method. In the future, we will focus on client subgraph sampling training in FL for scenarios where only a small fraction of clients participate. Additionally, we plan to apply graph-assisted FL to address more practical challenges in a real FL system, such as FL with noisy labels [23].

Acknowledgements

This work is supported by the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDC02030400.

References

[1] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan, 'A theory of learning from different domains', *Machine learning*, **79**(1), 151–175, (2010).

[2] James C Bezdek and Richard J Hathaway, 'Some notes on alternating optimization', in *AFSS international conference on fuzzy systems*, pp. 288–300. Springer, (2002).

[3] Christopher Briggs, Zhong Fan, and Peter Andras, 'Federated learning with hierarchical clustering of local updates to improve training on non-iid data', in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE, (2020).

[4] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar, 'Leaf: A benchmark for federated settings', *arXiv preprint arXiv:1812.01097*, (2018).

[5] Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He, 'Federated meta-learning with fast convergence and efficient communication', *arXiv preprint arXiv:1802.07876*, (2018).

[6] Fengwen Chen, Guodong Long, Zonghan Wu, Tianyi Zhou, and Jing Jiang, 'Personalized federated learning with graph', *arXiv preprint arXiv:2203.00829*, (2022).

[7] Fengwen Chen, Shirui Pan, Jing Jiang, Huan Huo, and Guodong Long, 'Dagcn: dual attention graph convolutional networks', in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, (2019).

[8] Yu Chen, Lingfei Wu, and Mohammed Zaki, 'Iterative deep graph learning for graph neural networks: Better and robust node embeddings', *Advances in neural information processing systems*, **33**, 19314–19326, (2020).

[9] Corinna Cortes and Mehryar Mohri, 'Domain adaptation and sample bias correction theory and algorithm for regression', *Theoretical Computer Science*, **519**, 103–126, (2014).

[10] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi, 'Adaptive personalized federated learning', *arXiv preprint arXiv:2003.13461*, (2020).

[11] Canh T Dinh, Tung T Vu, Nguyen H Tran, Minh N Dao, and Hongyu Zhang, 'Fedu: A unified framework for federated multi-task learning with laplacian regularization', *arXiv preprint arXiv:2102.07148*, **400**, (2021).

[12] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar, 'Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach', *Advances in Neural Information Processing Systems*, **33**, 3557–3568, (2020).

[13] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi, 'Slaps: Self-supervision improves structure learning for graph neural networks', *Advances in Neural Information Processing Systems*, **34**, 22667–22681, (2021).

[14] Xingbo Fu, Binchi Zhang, Yushun Dong, Chen Chen, and Jundong Li, 'Federated graph machine learning: A survey of concepts, techniques, and applications', *ACM SIGKDD Explorations Newsletter*, **24**(2), 32–47, (2022).

[15] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran, 'An efficient framework for clustered federated learning', *Advances in Neural Information Processing Systems*, **33**, 19586–19597, (2020).

[16] Alex Graves and Alex Graves, 'Long short-term memory', *Supervised sequence labelling with recurrent neural networks*, 37–45, (2012).

[17] Otkrist Gupta and Ramesh Raskar, 'Distributed learning of deep neural network over multiple agents', *Journal of Network and Computer Applications*, **116**, 1–8, (2018).

[18] Will Hamilton, Zhitao Ying, and Jure Leskovec, 'Inductive representation learning on large graphs', *Advances in neural information processing systems*, **30**, (2017).

[19] Filip Hanzely and Peter Richtárik, 'Federated learning of a mixture of global and local models', *arXiv preprint arXiv:2002.05516*, (2020).

[20] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons, 'The non-iid data quagmire of decentralized machine learning', in *International Conference on Machine Learning*, pp. 4387–4398. PMLR, (2020).

[21] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang, 'Personalized cross-silo federated learning on non-iid data.', in *AAAI*, pp. 7865–7873, (2021).

[22] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo, 'Semi-supervised learning with graph learning-convolutional networks', in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11313–11320, (2019).

[23] Xuefeng Jiang, Sheng Sun, Yuwei Wang, and Min Liu, 'Towards federated learning against noisy labels via local self-regularization', in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 862–873, (2022).

[24] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan, 'Improving federated learning personalization via model agnostic meta learning', *arXiv preprint arXiv:1909.12488*, (2019).

[25] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al., 'Advances and open problems in federated learning', *Foundations and Trends® in Machine Learning*, **14**(1–2), 1–210, (2021).

[26] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh, 'Scaffold: Stochastic controlled averaging for federated learning', in *International Conference on Machine Learning*, pp. 5132–5143. PMLR, (2020).

[27] Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar, 'Adaptive gradient-based meta-learning methods', *Advances in Neural Information Processing Systems*, **32**, (2019).

[28] Thomas N Kipf and Max Welling, 'Semi-supervised classification with graph convolutional networks', *arXiv preprint arXiv:1609.02907*, (2016).

[29] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon, 'Federated learning: Strategies for improving communication efficiency', *arXiv preprint arXiv:1610.05492*, (2016).

[30] A Krizhevsky, 'Learning multiple layers of features from tiny images', *Master's thesis, University of Tront*, (2009).

[31] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant, '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).

[32] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE*, **86**(11), 2278–2324, (1998).

[33] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith, 'Ditto: Fair and robust federated learning through personalization', in *International Conference on Machine Learning*, pp. 6357–6368. PMLR, (2021).

[34] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith, 'Federated learning: Challenges, methods, and future directions', *IEEE Signal Processing Magazine*, **37**(3), 50–60, (2020).

[35] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith, 'Federated optimization in heterogeneous networks', *Proceedings of Machine Learning and Systems*, **2**, 429–450, (2020).

[36] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu, 'Diffusion convolutional recurrent neural network: Data-driven traffic forecasting', *arXiv preprint arXiv:1707.01926*, (2017).

[37] Rui Liu and Han Yu, 'Federated graph neural networks: Overview, techniques and challenges', *arXiv preprint arXiv:2202.07256*, (2022).

[38] Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan, 'Towards unsupervised deep graph structure learning', in *Proceedings of the ACM Web Conference 2022*, pp. 1392–1403, (2022).

[39] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh, 'Three approaches for personalization with applications to federated learning', *arXiv preprint arXiv:2002.10619*, (2020).

[40] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh, 'Domain adaptation: Learning bounds and algorithms', *arXiv preprint arXiv:0902.3430*, (2009).

[41] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas, 'Communication-efficient learning of deep networks from decentralized data', in *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, (2017).

[42] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh, 'Agnostic federated learning', in *International Conference on Machine Learning*, pp. 4615–4625. PMLR, (2019).

[43] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, 'Mobilenetv2: Inverted residuals and linear bottlenecks', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, (2018).

[44] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek, 'Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints', *IEEE transactions on neural networks and learning systems*, **32**(8), 3710–3722, (2020).

[45] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar, 'Federated multi-task learning', *Advances in neural information processing systems*, **30**, (2017).

[46] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio, 'Graph attention networks', *arXiv preprint arXiv:1710.10903*, (2017).

[47] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar, 'Split learning for health: Distributed deep learning without sharing raw patient data', *arXiv preprint arXiv:1812.00564*, (2018).

[48] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol, 'Extracting and composing robust features with denoising autoencoders', in *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, (2008).

[49] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage, 'Federated evaluation of on-device personalization', *arXiv preprint arXiv:1910.10252*, (2019).

[50] Ruijia Wang, Shuai Mou, Xiao Wang, Wanpeng Xiao, Qi Ju, Chuan Shi, and Xing Xie, 'Graph structure estimation neural networks', in *Proceedings of the Web Conference 2021*, pp. 342–353, (2021).

[51] Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, Jing Jiang, and Chengqi Zhang, 'Multi-center federated learning', *arXiv preprint arXiv:2108.08647*, (2021).

[52] Pengwei Xing, Songtao Lu, Lingfei Wu, and Han Yu, 'Big-fed: Bilevel optimization enhanced graph-aided federated learning', *IEEE Transactions on Big Data*, (2022).

[53] Donghan Yu, Ruohong Zhang, Zhengbao Jiang, Yuexin Wu, and Yiming Yang, 'Graph-revised convolutional network', in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part III*, pp. 378–393. Springer, (2021).

[54] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra, 'Federated learning with non-iid data', *arXiv preprint arXiv:1806.00582*, (2018).

[55] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Yuanqi Du, Jieyu Zhang, Qiang Liu, Carl Yang, and Shu Wu, 'A survey on graph structure learning: Progress and opportunities', *arXiv e-prints*, arXiv–2103, (2021).

[56] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang, 'Deep graph structure learning for robust representations: A survey', *arXiv preprint arXiv:2103.03036*, **14**, (2021).