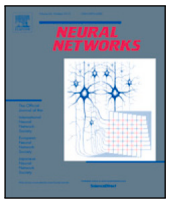
Neural Networks 189 (2025) 107541

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



Full Length Article

FedPPD: Towards effective subgraph federated learning via pseudo prototype distillation

Qi Lin [a,1], Jishuo Jia [a,1], Yinlin Zhu [b], Xunkai Li [c], Bin Jiang [a,*], Meixia Qu [a]

[a] Shandong University, School of Mechanical, Electrical and Information Engineering, Weihai, 264209, China
[b] Sun Yat-sen University, School of Computer Science and Engineering, Guangzhou, 510275, China
[c] Beijing Institute of Technology, School of Computer Science and Technology, Beijing, 100081, China

ARTICLE INFO

Keywords:
Subgraph federated learning
Data-free knowledge distillation
Semi-supervised node classification

ABSTRACT

Subgraph federated learning (subgraph-FL) is a distributed machine learning paradigm enabling cross-client collaborative training of graph neural networks (GNNs). However, real-world subgraph-FL scenarios often face subgraph heterogeneity problem, i.e., variations in nodes and topology across multiple subgraphs. As a result, the global model experiences a decline in performance. Despite several well-designed methods being proposed, most still rely on parameter aggregation-based global GNN for inference, which oversimplifies the subgraph knowledge and leads to sub-optimal performance. To this end, we propose achieving effective subgraph **fed**erated learning via **p**seudo **p**rototype **d**istillation (FedPPD). Specifically, FedPPD first utilizes a generator under the guidance of local prototypes to explore the global input space. Subsequently, the generated pseudo graph is used for distilling knowledge from the local GNNs to the vanilla-aggregated global GNN to convey reliable knowledge oversimplified during aggregation. Extensive experimental validation on six public datasets demonstrates that FedPPD consistently outperforms state-of-the-art baselines. Our code is available at https://github.com/KyrieLQ/FedPPD.

1. Introduction

Graph neural networks (GNNs) have emerged as the most popular paradigm for graph-based deep learning and demonstrated strong performance across various applications, including e-commerce recommendation systems (Wu, Sun et al., 2022), molecular structure analysis (Wang, Wang et al., 2022), and citation network categorization (Wu et al., 2021). However, most existing studies assume a centralized data storage setup, where the entire graph data is collected and accessed by a single institution. Unfortunately, this assumption often does not hold in real-world scenarios, where graph data is typically distributed across multiple institutions and is subject to local access restrictions. For example, each social media company maintains its own social network of registered users and wishes to collaborate on training a GNN for user classification. Due to user privacy policies and competitive business concerns, these companies are reluctant to share their private networks directly (Liu et al., 2022). In such scenarios, research on GNNs based on centralized settings becomes inadequate. This underscores the growing necessity for developing methods to train GNNs with collaborative intelligence in distributed environments.

To address this challenge, the concept of federated graph learning (FGL) has emerged, which combines the principles of federated learning (FL) to enable GNN training across decentralized scenarios. A notable variant of FGL is subgraph-FL, where each client retains an induced subgraph of an implicitly global graph and collaborates to execute node or edge-level downstream tasks (Zhang, Yang et al., 2021).

As a straightforward yet effective approach to implement subgraph-FL, FedAvg (McMahan et al., 2017) is originally developed for FL in the computer vision domain, which aggregates the global model by averaging the parameters from all local models. However, due to inherent differences in data sources and collection methods, there are significant variations in node features and graph topology arise across subgraphs, which have been shown to degrade the performance of the global GNN, known as the *subgraph heterogeneity* problem (Zhu et al., 2024).

To address this issue, Fed-PUB (Baek et al., 2023) introduces a personalized aggregation strategy based on subgraph similarity measuring. AdaFGL (Li, Wu, Zhang, Sun et al., 2024) conducts personalized local fine-tune based on local subgraphs and federated knowledge

* Corresponding author.
E-mail addresses: 202100800115@mail.sdu.edu.cn (Q. Lin), jiajishuo@mail.sdu.edu.cn (J. Jia), jiangbin@sdu.edu.cn (B. Jiang).
[1] These authors contributed to the work equally.

https://doi.org/10.1016/j.neunet.2025.107541
Received 16 August 2024; Received in revised form 4 February 2025; Accepted 23 April 2025
Available online 8 May 2025

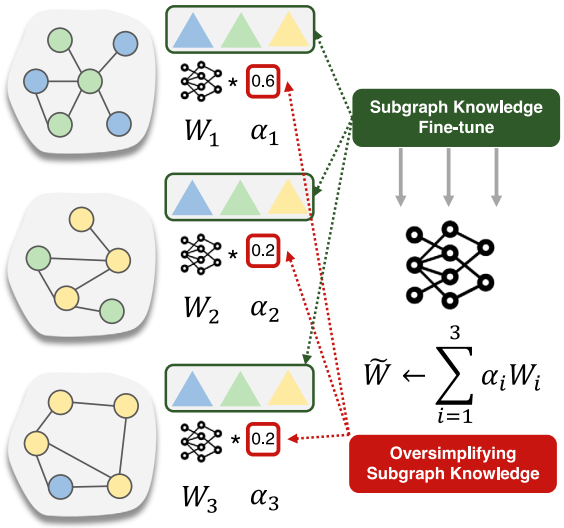


Fig. 1. An illustration of the subgraph knowledge oversimplifying problem caused by vanilla parameter aggregation. An intuitive solution is to fine-tune the global model using subgraph knowledge. W represents the model parameters of the local client, while α denotes the aggregation weight.

extractor. FedGTA (Li, Wu, Zhang, Zhu et al., 2024) employs topology-aware aggregation through local smoothing confidence and hybrid neighbor moments. Despite their effectiveness, these methods have a common *Limitation 1:* Relying on an aggregation-based global model for inference, which oversimplifies the subgraph knowledge and leads to sub-optimal performance. As illustrated in Fig. 1, during model aggregation, the node features and topological structures of each local subgraph are encoded into a single value (i.e., aggregation weight), losing essential local subgraph information and oversimplify the complex heterogeneity relationships across multiple local subgraphs.

Building on this insight, FedTAD (Zhu et al., 2024) further fine-tune the vanilla-aggregated global model through topology-aware data-free knowledge distillation. However, it suffers from *Limitation 2:* lacking sufficiently reliable guidance during fine-tuning. The optimization of generator in FedTAD is relies entirely on accurate predictions of local models on the pseudo graph, lacking more reliable guidance. As a result, the poor quality of the pseudo graph generated by FedTAD undermines the effectiveness of the distillation process.

To tackle these two limitations, we propose achieving **fed**erated learning through **p**seudo **p**rototype **d**istillation (FedPPD), which improves the performance of the vanilla-aggregated global model by leveraging local prototypes as a reliable local subgraph knowledge. Specifically, on the *Client side*, each client computes local prototypes to capture the nodes and topological information of its local subgraph. On the *Server side*, FedPPD uses a generator to produce a pseudo graph. Both the global model and local models use the pseudo graph to generate pseudo prototypes. By aligning pseudo local prototypes with real local prototypes, we obtain a well-trained generator. By aligning pseudo global prototypes with pseudo local prototypes, we fine-tune the global model to convey reliable knowledge oversimplified during aggregation. Notably, FedPPD is orthogonal to most model aggregation-based FGL algorithms, making it a hot-pluggable method for performance enhancement.

In summary, the contributions of this work can be summarized as follows:

- *Novel Method.* We introduce FedPPD, a new federated graph learning approach that addresses the oversimplification of knowledge in parameter aggregation. Instead of relying solely on traditional model averaging, FedPPD leverages prototypes to fine-tune the global model via data-free knowledge distillation. This mechanism effectively preserves important local structural information while ensuring data privacy.
- *State-of-the-Art Performance.* We conduct extensive experiments on six benchmark datasets, covering various graph learning tasks and data distributions. The experimental results demonstrate that FedPPD consistently outperforms existing state-of-the-art federated graph learning methods by a clear margin. These improvements validate the robustness and generalizability of our method across diverse scenarios.
- *Hot-Plugging Capability.* We show that FedPPD can be seamlessly integrated into many other FGL frameworks without requiring major architectural changes or additional training data. By simply incorporating the prototype-based distillation step, FedPPD effectively boosts performance and provides a flexible plugin option for existing methods, making it a practical choice in real-world federated graph applications.

Table 1
Description of notations commonly used in this paper.

Notations	Definitions
G	Original graph data
$\mathcal{V}$	Node set
$\boldsymbol{X}$	Node feature matrix
$\mathcal{E}$	Adjacency matrix
$\boldsymbol{Y}$	Label matrix
K	Number of clients
$\mathcal{D}_k$	dataset of k-client
ω	Global model parameters
ω_k	k-client model parameters
θ	generator parameters
$\mathcal{P}$	real prototype
$\tilde{\mathcal{P}}$	pseudo prototype
$M(t)$	Set of clients selected for round t

2. Preliminaries

2.1. Notation definitions

Given an attributed graph $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, where $\mathcal{V}$ denotes the node set, $\mathcal{E}$ denotes the adjacency matrix, and $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \in \mathbb{R}^{N \times F}$ is the node feature matrix, $\boldsymbol{x}_i \in \mathbb{R}^F$ is the F-dimensional feature vector of node v_i, and $N = |\mathcal{V}|$ holds. Besides, $\boldsymbol{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N\} \in \mathbb{R}^N$ is the label matrix, $\boldsymbol{y}_i \in \mathbb{R}^{\mathcal{Y}}$ is a one-hot vector and $\mathcal{Y}$ represents the number of the classes. Further definitions of notations commonly used in the paper is shown in Table 1.

2.2. Problem formulation

Let ω be the model parameter in the server and ω_k in the kth client. In the work, there are K clients, where $\mathcal{D}_k = \left\{\left(x_k^i, y_k^i\right)\right\}_{i=1}^{N_k}$ is the dataset stored in k-th client, N_k is the corresponding number of samples. Each client possesses a distinct subgraph. Overall, federated graph learning can be formulated as the following problem:

$$\begin{aligned} &\min_{\omega} \frac{1}{K} \sum_{k=1}^{K} f(\omega_k), \\ &f(\omega_k) = -\frac{1}{N_k} \sum_{i=1}^{N_k} \mathcal{L}\left(x_k^i, y_k^i, E_k; \omega_k\right), \\ &\mathcal{L}\left(x_k^i, y_k^i, E_k; \omega_k\right) = \sum_{i \in \mathcal{V}_l} \sum_{j} \mathrm{Y}_{ij} \log\left(\mathrm{softmax}(\hat{\mathrm{Y}})_{ij}\right), \end{aligned} \tag{1}$$

where $\mathcal{L}$ is the loss function which quantifies the difference between the predicted and actual results of the model, E_k is adjacency matrix for client k and dataset D_k for each $k \in \{1, 2, \ldots, K\}$ could be distributed heterogeneously. Limited by the privacy protection mechanism in FGL,

the server is unable to directly access the local data of clients. To solve Eq. (1), for each communication round t, existing methods send the global model ω to a random set of clients $M(t)$ and optimize it by $\min_{\omega} f_k(\omega)$, $k \in M(t)$. The server collects the local models $\{\omega_k\}_{k\in M(t)}$, and aggregates them by averaging the gradients to update the global model ω.

Unfortunately, due to heterogeneity issues between clients, there are significant differences between different local models. It is worth noting that the heterogeneity issue discussed here differs from the heterogeneity in heterogeneous networks (Shi et al., 2014). In the context of federated graph learning, heterogeneity refers to the differences in subgraph topological structures, as well as the heterogeneity in node or edge features. Thus, traditional gradient averaging could cause information loss in local models, and the updated global model has weak generalization ability which means it is difficult to fully utilize the training capabilities of local models. To solve this issue,we propose a new method to adjust the global model in Section 3, so that the global model can preserve the knowledge in local models and maintain their performance as much as possible.

2.3. Related works

2.3.1. Graph neural networks (GNNs)

Previous researches applied convolutional neural networks (CNNs) to graph by using spatial and spectral convolution methods (Bruna et al., 2014). However, the cost of computing the proposed convolutional operation and eigenvalue of a large matrix is enormous (Kipf and Welling, 2017). To avoid diagonalizing, fast Chebyshev polynomial approximation up to K-order was used (Hammond et al., 2011). GCN (Kipf and Welling, 2017) further simplifies the computation by confining the filters only to operate in first-order neighborhood of each node. However, the computational process relies on the Laplacian eigenbasis, which is inherently reliant on the graph's topology. Consequently, this reliance hinders generalization. To this end, GAT (Veličković et al., 2018) introduces attention mechanism by using masked self-attentional layer. Furthermore, in real-world scenarios, the topology of the graph may not be readily accessible. Thus, inferring graph structures from available information is of great significance. SLAPS (Fatemi et al., 2021) uses a generator which selects top k neighbors by employing KNN algorithm and cosine similarity, and updates these edges to infer a task-specific latent structure to mitigate the problem. Additionally, numerous alternative graph neural network architectures, such as SGC (Wu et al., 2019) and VGAE (Kipf and Welling, 2016), have demonstrated efficacy in practical applications.

2.3.2. Federated graph learning (FGL)

Federated learning (FL) (McMahan et al., 2017) is a novel distributed machine learning paradigm enabling collaborative training on distributed data while ensuring privacy-preserving. As a special instance of FL, federated graph learning (FGL) adapts these principles to graph-structured data. FGL can be categorized into two main types: (i) inter-graph federated learning; (ii) intra-graph federated learning (Zhang et al., 2021). Inter-graph federated learning involves a distributed learning paradigm where each client holds an entire graph dataset, while the server performs graph-level tasks. But in intra-graph federated learning, which aims to solve node-level tasks, clients possess partial, latent representations of the entire graph. Conventional federated learning overlooks the topological characteristics inherent in graphs, consequently exhibiting sub-optimal performance when applied to graph data. Considering the topological structure of the graph, FedSage+ (Zhang, Yang et al., 2021) employs a mechanism for generating absent neighbors to address the issue of missing links within local subgraphs. FED-PUB (Baek et al., 2023) introduces an innovative aggregation methodology predicated on the similarity among subgraph embeddings, aimed at mitigating data heterogeneity. FedGTA (Li, Wu, Zhang, Zhu et al., 2024) employs mixed moments of neighbor features to quantify the similarity of subgraphs and accomplish topology-aware aggregation. AdaFGL (Li, Wu, Zhang, Sun et al., 2024) employs knowledge distillation to perform personalized training process.

FGL can be effectively integrated with various other machine learning methodologies, including prototype learning (Zhang, Liu et al., 2024), knowledge distillation (Zhang et al., 2022), and others. Their application within graph data can yield significant enhancements in model performance.

2.3.3. Data-free knowledge distillation (DFKD)

Knowledge distillation (KD) is a technique that transfers knowledge from a well-trained model (i.e., teacher model) to a less-trained model (i.e., student model). (Hinton et al., 2015). However, substantial original datasets are required for knowledge distillation, which cannot be satisfied in distributed scenarios. To this end, DFKD generates pseudo data to supplement datasets. To meet the demand, DFAD-GNN (Zhuang et al., 2022) use two discriminators and a generator to infuse the knowledge into global model. CMI (Fang et al., 2021) combines DFKD with contrastive learning to address model collapse.

Integrating federated learning with data-free knowledge distillation has been shown to facilitate the maintenance of large federated models (Wu, Wu et al., 2022) and mitigate the phenomenon of global model drift caused by data heterogeneity (Zhu et al., 2021). Recent researches like FedGen and FedFTG (Zhang et al., 2022) introduces DFKD to FL scenarios. However, it should be noted that all of these methodologies are specifically tailored for Euclidean data and may not be directly applicable in the context of FGL.

2.3.4. Homogeneity and heterogeneity

The issue of homogeneity and heterogeneity has become a prominent topic in the fields of network science and graph analysis. In the domain of network science, heterogeneous information networks are conceptual structures composed of various types of objects and various types of links that represent distinct relationships, in contrast to homogeneous networks (Shi et al., 2014). Most recent studies treat practical systems as homogeneity information networks, overlooking distinctions between various types of entity and relationships within the networks. However, this modeling approach neglects the rich semantic and structural information inherent in the networks (Shi et al., 2017).

In the context of federated graph learning, the issue of homogeneity and heterogeneity refers to the consistency or disparity in the topological structures, node or edge attributes of graph data across different clients (Xie et al., 2024). The heterogeneity in FGL essentially reflects the topological divergence among multiple clients, specifically homophily or heterophily (Li, Wu, Zhang, Sun et al., 2024).

3. Proposed method

In this section, we describe the proposed FedPPD, which is depicted in Fig. 2. Concretely, on the *client side*, each client calculates its own local prototype using its local subgraph (i.e., real local prototype). Then the client sends three parts to the server, including: (i) real local prototype; (ii) model parameters; (iii) label distribution. On the *server side*, FedPPD utilizes a generator to explore the global input space and uses the generated pseudo graph to transfer knowledge from clients to the server. Then FedPPD fine-tunes the global model parameter through prototypes under the guidance of the pseudo graph to convey reliable knowledge oversimplified during aggregation. The corresponding algorithm is summarized in Algorithm 1.

3.1. Client side

Local prototype representation. To address the aggregation bias and sub-optimal performance issues of global GNN, our key insight is that

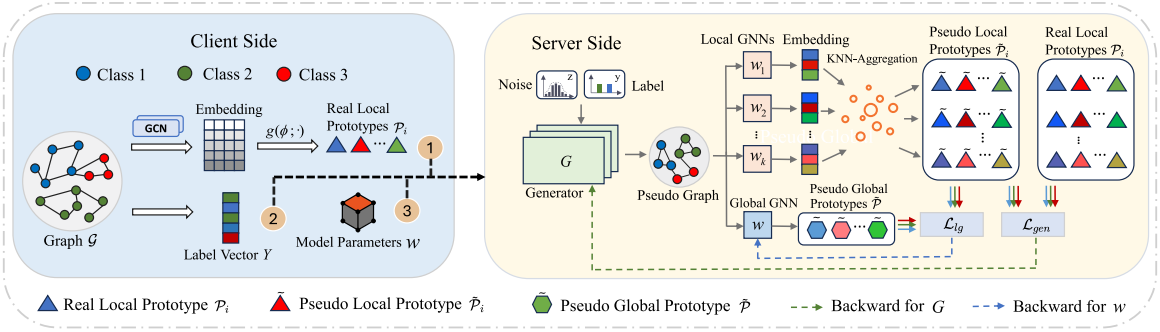


Fig. 2. An overview of proposed FedPPD. On the client side, each client initializes model parameters and collects knowledge by real local prototypes. On the server side, FedPPD integrates knowledge to generate a pseudo graph and further guides the optimization of global model, achieving reliable knowledge transfer.

using only model parameters leads to knowledge simplification, thus requiring a new knowledge carrier as training guidance. To better illustrate the features and topological information of subgraph nodes, we employ the real local prototype of the client as the information carrier. Prototype $\mathcal{P}^c$ represents the class c in all label classes C. Each client has a specific embedding function f_i. We denote $f_i(x)$ as the first-order embedding of node feature x for client i obtained by GCN. To further represent client information,

$$\mathcal{P}_i^c = \frac{1}{|S^c|} \sum_{(x_j, y_j) \in S^c} f_i(x_j), \tag{2}$$

is the prototype of class c in client i, where S^c means the set of real samples belonging to class c.

3.2. Server side

Knowledge generation. Due to the limitation of subgraph federated learning inability to share data, each client model is only trained by its own data, which weakens the generalization ability of the aggregated global model. In this case, we reconstruct the pseudo graph that is infinitely close to the original graph, as the output of knowledge aggregation on the server side, and apply them to the global GNN training process. As mentioned earlier, we propose a data-free knowledge distillation method, which means the server maintains a generator G that generates the pseudo graph to capture the data distribution of clients as follows:

$$\tilde{\mathbf{X}} = G(z, y; \theta), \tag{3}$$

where $\tilde{\mathbf{X}} \in \mathbb{R}^{\mathcal{U} \times F}$, $\mathcal{U}$ is the number of generated nodes, F is the dimension of generated node feature, θ is parameter of G, $z \sim \mathcal{N}(0, 1)$ is a standard Gaussian noise and y is the label of $\tilde{\mathbf{x}}$ sampled from predefined distribution $\psi(y)$. Then we construct the topology structure of the pseudo graph based on $\tilde{\mathbf{X}}$ with the $K - Nearest\ Neighbors$ strategy. Specifically, the pseudo adjacency matrix $\tilde{\mathcal{E}}$ is computed as follows:

$$\tilde{\mathcal{E}}[m, n] = \begin{cases} 1, & \text{if } m \in TopK(\mathbf{H}[m, :]), \mathbf{H} = \sigma(\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T); \\ 0, & \text{others}, \end{cases} \tag{4}$$

where $\sigma(\cdot)$ represents the sigmoid function. The obtained pseudo graph is denoted as $\tilde{G} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$ with the pseudo features $\tilde{\mathbf{X}}$ and the pseudo labels $\tilde{\mathbf{Y}}$.

Each round, selected clients calculate the real local prototype representation $\mathcal{P}_k$ and send it to server with both the local model parameter ω_k and label distribution y. After receiving the information from the client, The server firstly aggregate selected client model parameters into a preliminary global model $\boldsymbol{\omega}$ by objective function of FedAvg (McMahan et al., 2017) as follows:

$$\omega = \sum_{i=1}^{k} \frac{n_i}{n} \omega_i, \tag{5}$$

where k is the number of selected clients, n_i is the number of samples for client i and n is the total number of all selected clients in current epoch.

Knowledge distillation. The ultimate purpose of FedPPD for knowledge distillation is to achieve effective transfer of knowledge from clients to server, thereby fine-tuning the global model. For this purpose, the key insight is to generate samples that are highly consistent with real samples. We introduce a prototype-based optimization method of generator G. Formally, the overall collaborative training objective is

$$\min_{\theta} \mathcal{L}_{gen} = \frac{1}{|M(t)| \times C} \sum_{i \in M(t)} \sum_{c=1}^{C} \phi(\mathcal{P}_i^c, \tilde{\mathcal{P}}_i^c), \tag{6}$$

where $M(t)$ is the set of clients selected in current round, each c belongs to total number of classes C, $\mathcal{P}$ is the *real local prototype* transmitted by the client and $\tilde{\mathcal{P}}$ is the *pseudo local prototype* obtained by the pseudo graph via the local model parameters, $\phi(\cdot, \cdot)$ represents the Euclidean distance between two variables.

For a specific class c and client i, the pseudo prototype is denoted as $\tilde{\mathcal{P}}_i^c \in \mathbb{R}^d$ which is the representative of nodes belonging to class c in client i. We obtain that by averaging the first-order embedding of features in the same class:

$$\tilde{\mathcal{P}}_i^c = \frac{1}{|\tilde{S}^c|} \sum_{(\tilde{x}_j, \tilde{y}_j) \in \tilde{S}^c} f_i(\tilde{x}_j), \tag{7}$$

where $\tilde{S}^c$ means the set of pseudo samples in class c, $f_i(x)$ is the first-order embedding through GNN model in client i. We opt for first-order embedding primarily because the original GCN framework restricts the receptive field of each node to its immediate neighbors (one-hop). Adhering to this established principle not only preserves consistency with the foundational GCN design but also ensures that the local neighborhood structure is effectively captured without introducing additional model complexity. The initial pseudo prototype of each node is a typical representative of their respective semantic information. Restricted by the instability of the generator, obtaining the pseudo prototype solely from a single node is unreliable. We propose a neighbor protection mechanism to alleviate the issue of unreliable pseudo prototypes in this situation. Specifically, we assign weights to the initial pseudo prototype of a node and the prototype of its one hop neighbor node then aggregating them, the formula is as follows:

$$\begin{aligned} \tilde{\mathcal{P}}_i^c &= \frac{1}{|S^c|} \sum_{(\tilde{x}_j, \tilde{y}_j) \in S^c} g_i(\tilde{x}_j), \\ g_i(\tilde{x}_j) &= \alpha f_i(\tilde{x}_j) + \beta \mathbb{E}_{t \sim \chi(j)} f_i(\tilde{x}_t), \end{aligned} \tag{8}$$

where $\chi(j)$ represents one-hop neighbors of node j. Unlike the previous strategy that all nodes in the pseudo graph have the same number of neighbors, we allow each node to have a different number of neighbors, which not only enhances the authenticity and reliability of the pseudo graph structure, but also improves the similarity between adjacent nodes. FedPPD believes that when there is a significant difference between $g(\tilde{x}_j)$ and $f(\tilde{x}_j)$, neighbors will have a negative impact on the initial prototype representation. Therefore, we will reduce neighbors to alleviate this unfavorable factor.

Our goal in optimizing the generator is to enable the pseudo graph to better describe the data state of clients and provide sufficient client knowledge for using its local pseudo prototypes. In this context, we represent the global model in form of global pseudo prototype. To facilitate effective knowledge transfer, we utilize pseudo local global prototype loss L_{lg} in Eq. (9). By minimizing L_{lg}, knowledge transfers from local models to the global model.

$$\mathcal{L}_{lg} = \frac{1}{|M(t)| \times C} \sum_{i \in M(t)} \sum_{c=1}^{C} \phi(\tilde{\mathcal{P}}_i^c, \tilde{\mathcal{P}}_g^c), \tag{9}$$

where $\tilde{\mathcal{P}}_g$ is a pseudo global prototype obtained by propagating the pseudo graph forward through the global model.

In general, the entire training process on the server side can be formulated as an adversarial learning scheme:

$$\min_{\boldsymbol{w}} \max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{z} \sim \mathcal{N}(\mathbf{0},\mathbf{1}), \boldsymbol{y} \sim \psi(y)} [\mathcal{L}_{lg} - \mathcal{L}_{gen}]. \tag{10}$$

Algorithm 1 FedPPD Execution on Client and Server

Input: attributed graph $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, communication rounds T, client numbers K, client participating proportion ξ, generation iteration I_g, training iteration I_t, server training rounds I;

for each communication round $t = 1, ..., T$ **do**
  Select clients for round t, $M(t) = \{\text{random set of } \lceil \xi \times K \rceil \text{ clients}\}$;
  Client side:
  Update local model $\{\omega_k\}_{k \in M(t)} = \omega$;
  Calculate real local prototype according to Eq. (7);
  Server side:
  Aggregate local models ω_i according to Eq. (5);
  Label sampling and generating noise (z, y);
  for each server training round $i = 1, ..., I$ **do**
    for each training iteration $i_g = 1, ..., I_g$ **do**
      Generate a pseudo graph $\tilde{G}$ according to Eqs. (3),(4);
      Calculate and update pseudo local prototypes according to Eqs. (7),(8);
      Update generator parameter θ according to Eq. (6);
    end for
    for each generation iteration $i_t = 1, ..., I_t$ **do**
      Update global model parameter ω according to Eq. (9);
    end for
  end for
end for

4. Experiments

In this section, we provide a detailed introduction to a series of experiments used to explore the effectiveness of proposed FedPPD. Our experiment has been demonstrated from multiple perspectives to answer the following questions: **Q1**: Compared with other state-of-the-art FGL studies, can FedPPD achieve better performance? **Q2**: Where is the necessity of each component in FedPPD reflected? **Q3**: Is FedPPD sensitive to the hyperparameters? **Q4**: What are the underlying reasons for the effectiveness of our proposed method? **Q5**: As a dynamic hot-plugging method, what level of performance enhancement can FedPPD bring?

4.1. Experimental setup

Datasets. We perform experiments using six benchmark datasets of different scales to make our experimental results more objective and convincing. These include two smaller citation networks, Cora and Citeseer (Yang et al., 2016); a medium-sized citation network PubMed (Yang et al., 2016); dense co-purchase and co-author networks, Amazon-Computers and Coauthor Physics (Shchur et al., 2019); a large citation network ogb-arxiv (Hu et al., 2020). More details can be found in Table 2. According to above datasets, we utilize Louvain algorithm (Blondel et al., 2008) which is extensively utilized in FGL (Wang, Kuang et al., 2022) to emulate distributed scenarios in subgraph-FL.

Table 2
Statistics of the six public benchmark datasets.

Dataset	#Nodes	#Features	#Edges	#Classes
Cora	2,708	1,433	5,278	7
CiteSeer	3,327	3,703	4,552	6
PubMed	19,717	500	44,324	3
Amazon-Computers	13,752	767	245,861	10
Coauthor Physics	34,493	8,415	247,962	5
ogb-arxiv	169,343	128	1,166,243	40

Baselines. We compare the proposed FedPPD with different FL optimization strategies, including three traditional FL optimization strategies (FedAvg (McMahan et al., 2017), FedProx (Li et al., 2020) and Moon (Li et al., 2021)), two personalized subgraph FL optimization strategies (Fed-PUB (Baek et al., 2023) and FedGTA (Li, Wu, Zhang, Zhu et al., 2024)), a knowledge distillation subgraph FL method (FedTAD (Zhu et al., 2024)), two FL optimization strategies for client heterogeneity (FedProto (Tan et al., 2022) and FedTGP (Zhang, Liu, Hua and Cao, 2024)).

Hyperparameter settings. Unless otherwise specified, we use the following configuration for all algorithms. We use a two-layer GCN as our basic model with the hidden layer dimension set to 64 and the learning rate set to 1e-3. We set the number of clients to 10, the communication rounds to 100. The dropout, weight decay, batch size and the optimizer are set to 0.5, 1e-5, 128 and Adam respectively. For the generator, we set the number of generated nodes in the pseudo graph to 140 and the initial value of the number of neighbors K selected by the $K - NN$ algorithm is set to 5. To ensure the stability of the generated pseudo graph, the number of edges for each node in the pseudo graph cannot be reduced to less than 2. Then we conduct automated hyperparameters tuning by using Optuna toolkit (Akiba et al., 2019). We choose random hyperparameters search strategy to optimize our method. Below is the meaning of hyperparameters we searched for: (i) the training round of the server-side generator I_g, (ii) training round of global parameter I_t, (iii) the learning rate of aligning the pseudo global prototypes towards the pseudo local prototypes process lr_t (iv) the learning rate of the generator lr_g. The search space for the hyperparameters are : (i) I_g: from 1 to 10. (ii) I_t: from 1 to 10. (iii) lr_t: from 1e-6 to 1e-3. (iv) lr_g: from 1e-6 to 1e-3.

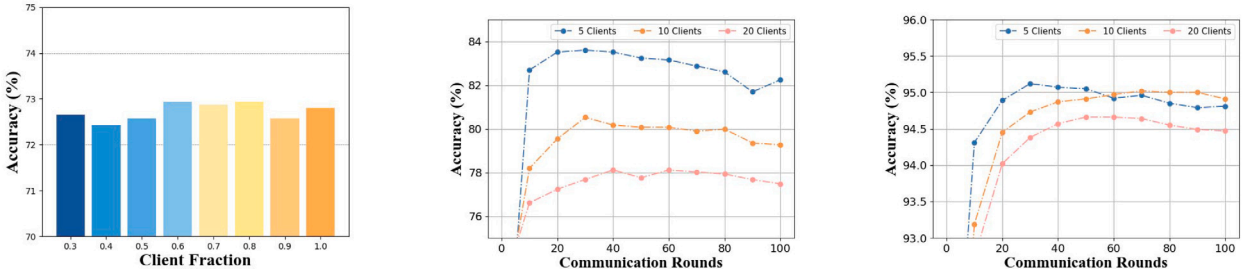
Experiment environment. We conducted our experiment on the computer with Intel(R) Core(TM) i9-14900HX CPU @ 2.20 GHz and NVIDIA GeForce RTX 4070 with 16 GB memory. The operation system is Windows 11.

4.2. Experimental results

To answer **Q1**, we compare the classification performance of different algorithms in this section on six datasets with 10 clients. As shown in the Table 3, the proposed FedPPD outperforms state-of-the-art baselines. Specifically, compared to FedAvg, it can achieve a performance prompt of up to 4.56%. And compared to the most competitive method FedTAD, FedPPD has also improved accuracy by 2.13%. Particularly, FedPPD increases by up to 10.06% compared to the FedProto on the CiteSeer dataset. We also test the accuracy of our method in different number of clients scenarios. We conduct the experiment on the Cora and Coauthor Physics dataset. The experiment result is shown in Fig. 3(b) and Fig. 3(c). The accuracy generally increases as the number of participating clients decreases. Meanwhile, we measure the accuracy under different client participation ratios in CiteSeer dataset, ranging from 0.3 to 1, with an interval of 0.1. As observed in Fig. 3(a), both excessive and insufficient participation ratios have a negative impact on the accuracy of the algorithms.

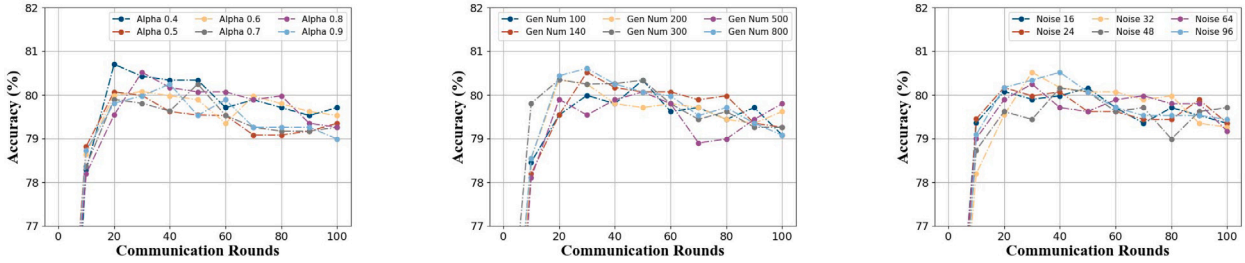
Table 3
Performance comparison of test accuracy achieved by FedPPD and baseline models on six datasets with ten clients. The best results are highlighted in **bold**, sub-optimal results are marked with an underline, and the third-best results are indicated by shading.

Method / Dataset	FedAvg	FedProx	MOON	Fed-PUB	FedGTA	FedTAD	FedProto	FedTGP	FedPPD (Ours)
Cora	75.87	75.96	76.68	78.46	78.74	78.38	77.75	77.39	**80.43**
CiteSeer	69.98	69.91	69.54	70.37	70.94	70.80	62.74	63.03	**72.80**
PubMed	83.65	83.84	83.46	84.87	84.45	85.17	83.79	83.79	**86.99**
Amazon-Computers	84.89	84.92	85.46	86.85	87.10	87.47	83.08	82.03	**89.27**
Coauthor Physics	91.62	91.61	91.65	92.91	92.96	93.19	91.72	91.60	**94.99**
obg-arxiv	65.15	65.08	65.10	66.78	66.58	66.58	62.23	64.29	**68.71**



(a) Different participation rate (b) Cora (c) Physics

Fig. 3. (a): Performance with different participating rate on CiteSeer dataset in 10 clients scenarios. (b) and (c): Convergence curves of different number of clients participating on Cora and Physics dataset.



(a) Alpha (b) Pseudo Graph Vertices (c) Noise Dimension

Fig. 4. Hyperparameters analysis with the training process of the proposed FedPPD.

4.3. Ablation study

To answer **Q2**, we conduct ablation experiments to explore the importance of each component. FedPPD consists of two main modules and an auxiliary module, including utilizing a generator under the guidance of local prototypes to explore the global input space, distilling knowledge from the local GNNs to the vanilla-aggregated global GNN and neighbor protection mechanism. After configuring the Cora and Amazon-Computer datasets, we conduct ablation experiment by discarding generator training process (w/o Gen. for short), neighbor prototypes aggregation process (w/o Neig. for short) and pseudo-global prototypes approaching pseudo-local prototypes process (w/o Appr. for short) respectively. The experiment result is shown in Table 4. After comparison, we can find that deleting any module will lead to a decrease in algorithm performance.

Table 4
Ablation Experiment Result on the Cora and Amazon-Computers datasets.

Datasets	Method	Accuracy (%)
Cora	FedPPD	80.43
	FedPPD (w/o Gen.)	79.99
	FedPPD (w/o Neig.)	79.81
	FedPPD (w/o Appr.)	79.80
Amazon-Computers	FedPPD	89.27
	FedPPD (w/o Gen.)	88.93
	FedPPD (w/o Neig.)	89.00
	FedPPD (w/o Appr.)	88.75

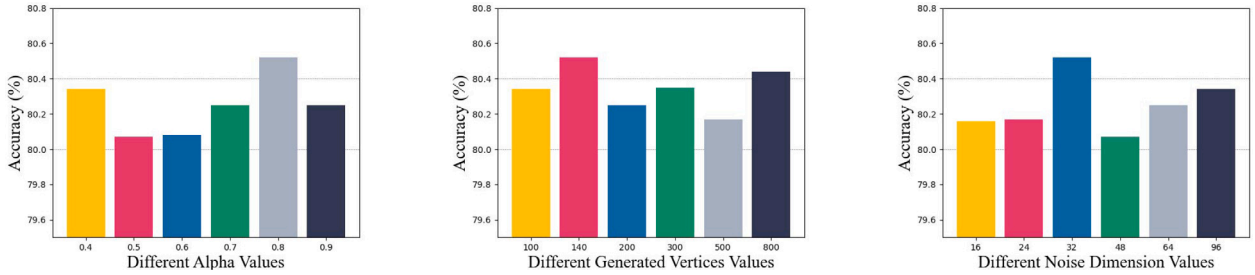
4.4. Hyperparameters sensitivity analysis

To answer **Q3**, we conduct the hyperparameters sensitivity analysis experiment on the Cora dataset using GCN as a regular GNN model to evaluate the accuracy of the model under different hyperparameter settings.

Firstly, to analyze whether the feature information of neighboring nodes will provide excessive guidance to this node, we change the aggregation ratio α between the neighboring prototype and this prototype to verify the difference in classification accuracy at different values. As shown in Fig. 4(a), with the change of alpha, the accuracy of the model remains stable between 80% and 81%, indicating that the model is not sensitive to alpha.

Secondly, we investigate the differences in models with different numbers of vertices in the generated pseudo graph. As shown in Fig. 4(b), when the number of vertices varies between 100 and 800, the accuracy difference in each round is still controlled within 2%, indicating that the number of vertices does not have a significant impact on the model.

Finally, we investigate the impact of different noise dimensions on model accuracy. As shown in Fig. 4(c), even at the location with



(a) Alpha (b) Pseudo Graph Vertices (c) Noise Dimension

Fig. 5. Performance comparison with different hyperparameters of our method FedPPD.

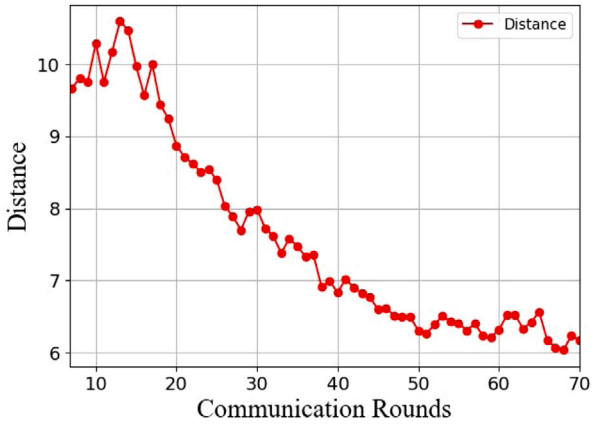


Fig. 6. Effectiveness analysis of the proposed method FedPPD on the Cora dataset.

the largest difference, the impact of 24 dimensional noise and 48 dimensional noise on the accuracy of the model is only 1%, indicating that the model is not sensitive to the noise dimension.

Simultaneously, we compare the highest test set accuracy achieved under different values of three hyperparameters to investigate the impact of hyperparameters on the optimal training results of the model. The result shown in Fig. 5 indicates that different parameter values have little effect on the training performance of the model.

In summary, FedPPD demonstrates low sensitivity to the aggregation ratio α, which governs the weighting between the prototype of neighboring nodes and its own prototype, as well as to the number of the generated pseudo graph vertices and the dimension of the generator's original input, thereby making the model highly robust.

4.5. Effectiveness analysis

To answer **Q4**, we analyze the reasons why FedPPD works. In FedPPD, the prototype is the carrier of knowledge, the optimization of server-side model parameters depends on the extent to which the pseudo local prototype approaches the real local prototype, which fundamentally depends on the quality of the pseudo graph generated by the generator. We record the distance between the global pseudo graph and the real graph at each training iteration. The result is shown in Fig. 6. Apparently, as training progresses, the distance between the global pseudo graph and the real graph decreases, indicating an improvement in the quality of the generated pseudo graph. It demonstrates the effectiveness of proposed FedPPD, as it indicates that the optimization of server-side parameters is successfully guided by the improved quality of the pseudo graph.

4.6. Hot-plugging performance enhancement

To answer **Q5**, we integrate FedPPD as a plugin into several FGL baseline methods, especially in the situation of subgraph-FL, including Fed-PUB and FedGTA. Since Fed-PUB and FedGTA both optimize the aggregation of federated parameters as well as the client-side update process, and given that our method primarily relies on server-side computation without personalized parameter aggregation, we can incorporate their personalized learning strategies and client-side procedures. Through this integration, we can fully leverage resources on both the client and server sides, thereby enhancing overall performance. We evaluate their performance improvement on the Cora dataset with ten participating clients. The experiment result is shown in Table 5. As observed, these two algorithms show a significant increase in accuracy after integration, which indicates that it can improve the performance when FedPPD is applied as a plugin to other algorithms.

Table 5
Performance improvement attributed to the integration of the FedPPD plugin on the Cora dataset.

Methods	w/o FedPPD	with FedPPD
Fed-PUB	78.46	79.98
FedGTA	78.74	80.25

5. Discussion

Privacy concerns. At each communication round, FedPPD need to upload the model parameters, label distribution and real local prototypes. The additional upload prototypes may raise privacy concerns. However, prototypes are high-dimensional representations of data, making it challenging to interpret their actual meaning. Moreover, the generator is trained by prototypes of the clients. Consequently, the generated pseudo graph tends to reveal the whole feature of the local subgraphs, and personalized features are ignored.

Limitations. In our methods, the client is responsible for calculating the real local prototypes based on the local dataset and subsequently uploading these prototypes to the server. The server side needs to generate the pseudo graph and guide the global model parameter to update. As a result, compared to other simple aggregation FL methods such as FedAvg, the server side needs to have a stronger load bearing capacity, and due to the training time of the generator, the server incurs additional time overhead.

6. Conclusion

This paper investigates the issue of performance degradation that arises from the naive aggregation of model parameters in the presence of subgraph heterogeneity. To this end, we propose FedPPD, which conveys reliable knowledge which is oversimplified during the aggregation process to generator by means of prototypes. Consequently, FedPPD can fine-tune the global model to a better performance. We consider the topological structure of graph by aggregating neighbor prototypes and own prototypes, which is proved to have better performance in subgraph-FL. Extensive experiments revealing the effectiveness of FedPPD, which outperforms various state-of-the-art baselines consistently.

CRediT authorship contribution statement

Qi Lin: Writing – review & editing, Writing – original draft, Methodology, Investigation. **Jishuo Jia:** Writing – review & editing, Writing – original draft, Visualization, Methodology, Investigation. **Yinlin Zhu:** Methodology, Investigation. **Xunkai Li:** Methodology, Investigation. **Bin Jiang:** Supervision. **Meixia Qu:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This paper is supported by the Shenzhen Fundamental Research Program, China under Grant JCYJ20230807094104009.

Data availability

Data will be made available on request.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2623–2631). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3292500.3330701.

Baek, J., Jeong, W., Jin, J., Yoon, J., & Hwang, S. J. (2023). Personalized subgraph federated learning. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, & J. Scarlett (Eds.), *Proceedings of machine learning research*: *202*, *Proceedings of the 40th international conference on machine learning* (pp. 1396–1415). PMLR, URL https://proceedings.mlr.press/v202/baek23a.html.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*(10), P10008.

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2014). Spectral networks and locally connected networks on graphs. arXiv:1312.6203.

Fang, G., Song, J., Wang, X., Shen, C., Wang, X., & Song, M. (2021). Contrastive model inversion for data-free knowledge distillation. arXiv:2105.08584.

Fatemi, B., El Asri, L., & Kazemi, S. M. (2021). SLAPS: Self-supervision improves structure learning for graph neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems*: *vol. 34*, (pp. 22667–22681). Curran Associates, Inc., URL https://proceedings.neurips.cc/paper_files/paper/2021/file/bf499a12e998d178afd964adf64a60cb-Paper.pdf.

Hammond, D. K., Vandergheynst, P., & Gribonval, R. (2011). Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, *30*(2), 129–150.

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv:1503.02531.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., & Leskovec, J. (2020). Open graph benchmark: Datasets for machine learning on graphs. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems*: *vol. 33*, (pp. 22118–22133). Curran Associates, Inc., URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fb60d411a5c5b72b2e7d3527cfc84fd0-Paper.pdf.

Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. arXiv:1611.07308.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. arXiv:1609.02907.

Li, Q., He, B., & Song, D. (2021). Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10713–10722).

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated optimization in heterogeneous networks. In I. Dhillon, D. Papailiopoulos, & V. Sze (Eds.), *2*, *Proceedings of machine learning and systems* (pp. 429–450). URL https://proceedings.mlsys.org/paper_files/paper/2020/file/1f5fe83998a09396ebe6477d9475ba0c-Paper.pdf.

Li, X., Wu, Z., Zhang, W., Sun, H., Li, R.-H., & Wang, G. (2024). AdaFGL: A new paradigm for federated node classification with topology heterogeneity. arXiv:2401.11750. URL https://arxiv.org/abs/2401.11750.

Li, X., Wu, Z., Zhang, W., Zhu, Y., Li, R.-H., & Wang, G. (2024). FedGTA: Topology-aware averaging for federated graph learning. arXiv:2401.11755.

Liu, Z., Yang, L., Fan, Z., Peng, H., & Yu, P. S. (2022). Federated social recommendation with graph neural network. *ACM Transactions on Intelligent Systems and Technology*, *13*(4).

McMahan, B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. y. (2017). Communication-efficient learning of deep networks from decentralized data. In A. Singh, & J. Zhu (Eds.), *Proceedings of machine learning research*: *vol. 54*, *Proceedings of the 20th international conference on artificial intelligence and statistics* (pp. 1273–1282). PMLR, URL https://proceedings.mlr.press/v54/mcmahan17a.html.

Shchur, O., Mumme, M., Bojchevski, A., & Günnemann, S. (2019). Pitfalls of graph neural network evaluation. arXiv:1811.05868. URL https://arxiv.org/abs/1811.05868.

Shi, C., Kong, X., Huang, Y., S. Yu, P., & Wu, B. (2014). HeteSim: A general framework for relevance measure in heterogeneous networks. *IEEE Transactions on Knowledge and Data Engineering*, *26*(10), 2479–2492.

Shi, C., Li, Y., Zhang, J., Sun, Y., & Yu, P. S. (2017). A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, *29*(1), 17–37.

Tan, Y., Long, G., LIU, L., Zhou, T., Lu, Q., Jiang, J., & Zhang, C. (2022). FedProto: Federated prototype learning across heterogeneous clients. *36*, In *Proceedings of the AAAI conference on artificial intelligence* (8), (pp. 8432–8440).

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. arXiv:1710.10903.

Wang, Z., Kuang, W., Xie, Y., Yao, L., Li, Y., Ding, B., & Zhou, J. (2022). FederatedScope-GNN: Towards a unified, comprehensive and efficient package for federated graph learning. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 4110–4120). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3534678.3539112.

Wang, Y., Wang, J., Cao, Z., & Barati Farimani, A. (2022). Molecular contrastive learning of representations via graph neural networks. *Nature Machine Intelligence*, *4*(3), 279–287.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4–24.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. (2019). Simplifying graph convolutional networks. In K. Chaudhuri, & R. Salakhutdinov (Eds.), *Proceedings of machine learning research*: *97*, *Proceedings of the 36th international conference on machine learning* (pp. 6861–6871). PMLR, URL https://proceedings.mlr.press/v97/wu19e.html.

Wu, S., Sun, F., Zhang, W., Xie, X., & Cui, B. (2022). Graph neural networks in recommender systems: A survey. *ACM Computing Surveys*, *55*(5).

Wu, C., Wu, F., Lyu, L., Huang, Y., & Xie, X. (2022). Communication-efficient federated learning via knowledge distillation. *Nature Communications*, *13*(1), 2032.

Xie, Z., Li, L., Chen, X., Yu, H., & Huang, Q. (2024). FedDGL: Federated dynamic graph learning for temporal evolution and data heterogeneity. In *Proceedings of machine learning research*: *vol. 260*, *Asian conference on machine learning, ACML 2024, 5-8 December, Hanoi, Vietnam* (pp. 1–16).

Yang, Z., Cohen, W., & Salakhudinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In M. F. Balcan, & K. Q. Weinberger (Eds.), *Proceedings of machine learning research*: *vol. 48*, *Proceedings of the 33rd international conference on machine learning* (pp. 40–48). New York, New York, USA: PMLR, URL https://proceedings.mlr.press/v48/yanga16.html.

Zhang, J., Liu, Y., Hua, Y., & Cao, J. (2024). FedTGP: Trainable global prototypes with adaptive-margin-enhanced contrastive learning for data and model heterogeneity in federated learning. arXiv:2401.03230.

Zhang, J., Liu, Y., Hua, Y., & Cao, J. (2024). FedTGP: Trainable global prototypes with adaptive-margin-enhanced contrastive learning for data and model heterogeneity in federated learning. *38*, In *Proceedings of the AAAI conference on artificial intelligence* (15), (pp. 16768–16776).

Zhang, L., Shen, L., Ding, L., Tao, D., & Duan, L.-Y. (2022). Fine-tuning global model via data-free knowledge distillation for non-IID federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10174–10183).

Zhang, H., Shen, T., Wu, F., Yin, M., Yang, H., & Wu, C. (2021). Federated graph learning – A position paper. arXiv:2105.11099.

Zhang, K., Yang, C., Li, X., Sun, L., & Yiu, S. M. (2021). Subgraph federated learning with missing neighbor generation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems*: *vol. 34*, (pp. 6671–6682). Curran Associates, Inc., URL https://proceedings.neurips.cc/paper_files/paper/2021/file/34adeb8e3242824038aa65460a47c29e-Paper.pdf.

Zhu, Z., Hong, J., & Zhou, J. (2021). Data-free knowledge distillation for heterogeneous federated learning. In M. Meila, & T. Zhang (Eds.), *Proceedings of machine learning research*: *vol. 139*, *Proceedings of the 38th international conference on machine learning* (pp. 12878–12889). PMLR, URL https://proceedings.mlr.press/v139/zhu21b.html.

Zhu, Y., Li, X., Wu, Z., Wu, D., Hu, M., & Li, R.-H. (2024). FedTAD: Topology-aware data-free knowledge distillation for subgraph federated learning. arXiv:2404.14061.

Zhuang, Y., Lyu, L., Shi, C., Yang, C., & Sun, L. (2022). Data-free adversarial knowledge distillation for graph neural networks. arXiv:2205.03811.