SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

January 2025, Vol. 68, Iss. 1, 112105:1–112105:15
https://doi.org/10.1007/s11432-024-4254-5

Personalized federated few-shot node classification

Yunfeng ZHAO[1,2], Xintong HE[3], Guoxian YU[1,2*], Jun WANG[1,2], Yongqing ZHENG[1,2] & Carlotta DOMENICONI[4]

[1]*School of Software, Shandong University, Jinan 250101, China;*
[2]*Joint SDU-NTU Centre for Artificial Intelligence Research, Shandong University, Jinan 250101, China;*
[3]*Department of Mathematics, National University of Singapore, Singapore 119077, Singapore;*
[4]*Department of Computer Science, George Mason University, Virginia 22030, USA*

Received 15 April 2024/Revised 17 August 2024/Accepted 25 September 2024/Published online 24 December 2024

Abstract Personalized federated learning (PFL) aims to train customized models for individual clients in a decentralized setting, with the account of non-independent and identically distributed data across clients. However, most PFL methods adopt uniform classification layers for diverse clients and give rise to error-prone predictions, due to the task heterogeneity notably prominent in decentralized graph data scenarios. Although some PFL solutions setup client-specific classification layers for each client and optimize them only locally, they are corrupted with limited local training data. We propose an innovative solution called federated parameter decoupling and node augmentation (`FedPANO`) to address these problems and to achieve personalized federated few-shot node classification, which is a prevalent and challenging but unexplored topic. Specifically, `FedPANO` first separates the local model into the GNN and classifier to handle unique client-specific task variations. The GNN is trained through federated learning to capture shared knowledge of graph nodes across clients, while the classifier is custom-designed and trained individually for each client. Additionally, a generic classifier shared among clients is adopted to encourage the GNN's grasp of shared information. Then `FedPANO` further proposes the node generator along with its local and collaborative training strategies to deal with the node scarcity of clients. Extensive experimental results on benchmark datasets confirm that `FedPANO` outperforms eight competitive baselines across different settings.

Keywords personalized federated learning, few-shot learning, node classification, task heterogeneity, parameter decoupling, node augmentation

Citation Zhao Y F, He X T, Yu G X, et al. Personalized federated few-shot node classification. Sci China Inf Sci, 2025, 68(1): 112105, https://doi.org/10.1007/s11432-024-4254-5

1 Introduction

To facilitate effective data mining from graph-structured data, numerous research has been made to develop graph neural networks (GNNs) [1, 2]. With impressive performance and generalizability, GNNs have been widely adopted to perform graph mining tasks (e.g., node classification [3]) across a variety of real-world domains, such as bio-medicine [4], knowledge graphs [5, 6], and social networks [7, 8]. A well-performed GNN typically needs a huge number of training graph data, whose nodes and edges are generally generated and collected from multiple edge devices (a.k.a., clients) in practice [9]. Unfortunately, due to the emphasis and regulations on data privacy (e.g., GDPR[1)]), data owners may not be willing or allowed to share the private local data for centralized model training [10–12], which significantly impair the model's performance.

Federated learning (FL) [10, 13, 14], as an emerging distributed learning paradigm, offers a promising framework for addressing privacy and security concerns by collaboratively training a model across disparate clients, while keeping their data within their local domains. It allows the establishment of robust models without centralizing the data, mitigating privacy and security risks associated with traditional centralized learning methods. Despite its effectiveness, a critical challenge in FL arises from the non-independent and identically distributed (non-i.i.d.) data among clients, which stems from clients' diverse behaviors and preferences [15–17]. The non-i.i.d. problem makes it challenging to fit the entire local data with a single global model, ultimately compromising the model's performance. To handle this problem,

* Corresponding author (email: gxyu@sdu.edu.cn)
1) https://gdpr-info.eu.

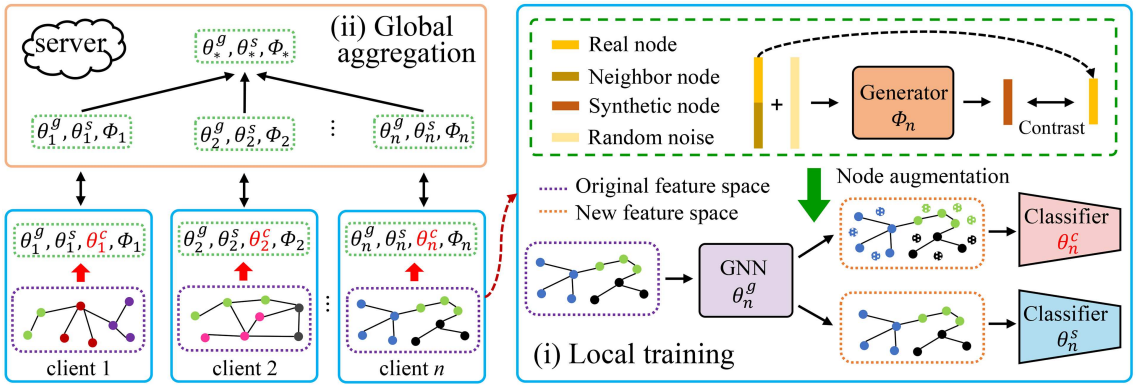

Figure 1 (Color online) Overall schematic framework of FedPANO. It iteratively performs local training and global aggregation steps. (i) Local training: clients decouple the local model into the GNN θ_i^g and client-specific classifier θ_i^c to handle task variations, and additionally adopt a classifier θ_i^s shared across clients to further enhance θ_i^g. Each client further handles the scarce client nodes via a meticulously designed node generator ϕ_i, after which each client c_i sends $\{\theta_i^g, \theta_i^s, \phi_i\}$ to the FL server. (ii) Global aggregation: the server first improves $\{\theta_i^g, \theta_i^s, \phi_i\}_{i=1}^n$ collaboratively with global aggregation, and then broadcasts the aggregated models to participating clients.

the personalized federated learning (PFL) [15, 18–20] framework was introduced to jointly induce a personalized model for each participating client, with the aim of these models better accommodating the non-i.i.d. local data. Another crucial problem for FL is that clients' samples are often interdependent (i.e., graph-structured data). Although several PFL approaches have been invented to deal with decentralized graphs [21–23], they typically incorporate uniform classification layers across different clients for training and inference, overlooking the task heterogeneity particularly conspicuous in decentralized graph data scenarios and giving rise to error-prone predictions.

To handle the task heterogeneity across clients, a canonically used manner is to divide the deep neural network into base and personalization layers, in which base layers are shared and aggregated in the server while personalization layers are kept private and optimized only with local data [24, 25]. However, this strategy builds on the promise that each client has sufficient training data to refine personalized layers. Consequently, they do not perform well in a few-shot scenario, where each client only holds a handful of training nodes for each label, due to costly annotations or dealing with uncommon tasks. Although few-shot graph learning (FSGL) has been explored in some domains [26, 27], the predominant nature of these approaches is centralized learning. This implies their reliance on centralized graphs to acquire meta-knowledge, thereby prohibiting their applicability to decentralized graphs.

Motivated by the above-mentioned problems, this paper focuses on personalized federated few-shot node classification, which holds wide applications. For instance, in a social network context, there is a growing demand to offer personalized recommendations to new users (few-shot data) with diverse behaviors and interests. These recommendations may include suggesting relevant communities, topics, or advertisements (task heterogeneity). Moreover, user data are bound with privacy concerns and direct data sharing in the network is not a viable option. This realistic but largely overlooked scenario presents two technical challenges. Challenge 1: How to deal with task heterogeneity among different clients? Solution 1: To address this challenge, we adopt parameter decoupling to decompose the local model into two parts: a GNN and a classifier. The GNN is trained within the FL framework, allowing the model to capture the shared and general knowledge across clients. Whereas the classifier is client-specific and trained only locally for each client, and thus can handle task heterogeneity. Besides, a generic classifier shared among clients is introduced to work in conjunction with GNN to further grapple with the general information. Challenge 2: How to improve the client-specific classifier with limited graph nodes? Solution 2: To overcome this challenge, we introduce a meticulously designed node generator with its local and collaborative training strategies, which considers the node connectivity to synthesize nodes for each client to further elevate the client-specific classifier.

The schematic framework of the proposed method is depicted in Figure 1, and the main contributions can be succinctly outlined as follows.

• We focus on a practical and challenging PFL problem, where clients have heterogeneous tasks but with few-shot training nodes. We formalize this problem as a new learning topic called personalized federated few-shot node classification and introduce the federated parameter decoupling and node augmentation

(FedPANO) method to approach it.

• FedPANO first decouples the local model into GNN and classifier to handle task variations among clients. It trains GNN within FL to capture shared insights across clients, additionally facilitated by a shared and generic classifier among clients, meanwhile managing a client-specific classifier on each client for task heterogeneity. Furthermore, it meticulously designs a node generator for each client with local and collaborative training strategies to handle scarce nodes.

• We conduct experiments using four benchmark datasets and compare our FedPANO against eight competitive methods [10, 21, 22, 24, 25, 28–30]. Extensive results demonstrate the effectiveness of FedPANO for dealing with task heterogeneity and scarce training graph nodes.

2 Related work

2.1 PFL

PFL [15, 20, 31, 32] has gained popularity for its advantage to overcome statistical heterogeneity among clients, which typically trains a personalized model for each participating client with the aim of the learned personalized models can better fit the corresponding local distributions. Most existing techniques for PFL can be roughly grouped into data-based and model-based methods [31].

The data-based techniques target to smooth the statistical heterogeneity among clients' datasets. To name a few, Yang et al. [33] proposed to select the subset of clients with minimal class imbalance, and revealed the local class distributions by comparing the similarity of local gradients with the gradients inferred from a balanced auxiliary dataset on the server. Duan et al. [34] proposed a self-balancing FL framework to handle imbalanced data among different clients by Z-score-based data augmentation and down-sampling of local data. Model-based strategies develop models to fit the various distributions of participating clients. Among them, a considerable amount of research has been dedicated to exploring diverse model aggregation strategies. For example, FedFomo (PFL with first-order model optimization) [35] learns optimal weights for model aggregation w.r.t. each client by figuring out how much a client can benefit from other clients. While CFL (clustering federated learning) [36] uses cosine similarity between updated weights of different clients as the similarity of their data distributions. Subsequently, it divides participating clients into multiple clusters based on this similarity measure, followed by executing FL within each cluster. pFedLA (PFL with layer-wised model aggregation) [37] evaluates the importance of each layer from different clients and achieves personalized model aggregation at the layer level. Baek et al. [22] proposed federated personalized subgraph learning (FED-PUB) for decentralized graph data. FED-PUB first estimates similarities between clients using functional embeddings of their models on random graphs and then employs these similarities to perform weighted model aggregation for clients.

However, these PFL approaches typically incorporate uniform classification layers for different clients to perform inference, overlooking the task heterogeneity among clients, which could lead to error-prone predictions. To address this, Arivazhagan et al. [24] and Collins et al. [25] proposed FedPer and FedRep, which divide the deep neural network into base and personalization layers. The base layers are shared and aggregated in the server, while the personalization ones are kept private and optimized only with local data. A dilemma for them is that they rely on clients having sufficient data to induce the personalized layers. As a result, they cannot perform well in a few-shot scenario, where clients only possess a limited number of training nodes per category due to costly annotations or dealing with uncommon tasks. Recently, some research has investigated FL with few-shot local data. For example, Zhao et al. [29] proposed pFedFSL (personalized federated few-shot learning) that leverages prototype representation learning to reduce dependence on sample size. pFedFSL first learns a personalized and discriminative embedding network for each client and then classifies testing samples according to their distance to labeled samples in the embedding feature space. While Wang et al. [30] proposed F^2L (federated few-shot learning) to address global data variance by decoupling the acquisition of local meta-knowledge, and subsequently harness the collective global insights gleaned from all clients to address local data insufficiency challenges. However, these methods neglect the interdependence between nodes, and thus suffer a compromising performance. Besides, Zhang et al. [21] investigated the missing links across different graphs within the client and introduced the FedSage+, in which each client first holds out some nodes and related links randomly, and subsequently trains a missing neighbor generator based on the held-out neighbors, which incurs difficulty for FedSage+ to utilize scarce nodes to train the neighbor generator. Furthermore, alike

most other PFL methods [22,37], it still adopts uniform classification layers for different clients to perform inference, which further degenerates the model's performance.

To tackle the above problems, our `FedPANO` first adopts parameter decoupling to tackle task heterogeneity, and then meticulously designs a node generator considering node connectivity, along with its local and collaborative training strategies to deal with scarce training nodes.

2.2 FSGL

FSGL [26], a sub-branch of few-shot learning [38–40] inspired by the observed human cognitive capability, aims to enable graph representation learning by leveraging a handful of annotated data. Typical FSGL problems encompass three primary graph mining tasks: few-shot node classification (FSNC) [41], few-shot relation prediction [42], and few-shot graph classification [43]. Given the importance of node classification in graph mining, we mainly focus on the realm of FSNC in this study. Existing FSNC problems are typically supervised ones, and they can be viewed as N-way K-shot node classification problems, in which the training dataset for a new task consists of $N \times K$ nodes, where N is the number of classes, and each class has K nodes. Approaches for FSNC can be broadly categorized into metric-based and optimization-based methods [26]. The former class of methods leverages their prior learned knowledge to quickly adapt to new tasks by learning an informative feature space, which can be used to predict based on input similarity scores [44, 45]. For example, Ding et al. [44] pioneered the exploration of node importance through the integration of a self-attention mechanism. Subsequently, they calculated the weighted summation of embeddings from support nodes, yielding a sophisticated prototype for each class. In a parallel vein, Lan et al. [45] harnessed an embedding transformation function to facilitate the mapping of task-agnostic node representations to their task-specific counterparts. This approach delves into the intricate and multifaceted relationships inherent in the interplay between nodes. The second category of methods aims at learning a well-initialized model, which can be quickly fine-tuned or adapted to perform well on new tasks with only a few nodes [46, 47]. For instance, Wang et al. [46] introduced an attribute-level attention mechanism, which aims at enhancing the modeling of distinctive properties within each meta-learning task. Additionally, Huang and Zitnik [47] provided theoretical justification asserting that evidence supporting a prediction can be discerned within the local subgraph proximal to the target node, and leveraged this subgraph to transfer subgraph-specific information and learn transferable knowledge faster via meta-gradients.

Unfortunately, these FSNC methods acquire meta-knowledge on the requisite of centralized graph nodes, and thus remains challenging to conduct FSNC under the FL setting, especially with global data variance. Our `FedPANO` adeptly leverages the shared GNN and node generator to acquire meta-knowledge from different clients with decentralized non-i.i.d. data, meanwhile preserving the specificity of individual clients via the client-specific classifiers.

3 Methodology

3.1 Problem formulation and notation

Suppose the FL system has n clients $\{c_i\}_{i=1}^n$, and each client c_i owns a graph dataset $G_i = \{\mathcal{V}_i, \mathcal{E}_i\}$ that are private and prohibited from sharing with the server and peers. Specifically, $\mathcal{V}_i$ is the set of nodes where each node u possesses a feature vector $\boldsymbol{x}_u \in \mathcal{X}$ and one-hot vector $\boldsymbol{y}_u \in \mathcal{Y}_i$, where $\mathcal{X}$ and $\mathcal{Y}_i$ denote the corresponding feature space and label space of $\mathcal{V}_i$, respectively. While $\mathcal{E}_i$ is the set of edges with $(u, v) \in \mathcal{E}_i$ representing an edge between node u and node v. Due to the diverse usages and preferences of clients, the label spaces of those clients are generally different from each other, i.e., $\mathcal{Y}_i \neq \mathcal{Y}_j$, causing task heterogeneity, also termed as label distribution skew, which is a typical non-i.i.d. scenario in FL [31]. In addition, the labeled data in each client are typically scarce since the costly annotations, i.e., each client only has K labeled nodes per category for training ($|\mathcal{Y}_i|$-way K-shot). Under such scenarios, we aim to collaboratively train a personalized model for each client to achieve accurate node classification. The framework of `FedPANO` is depicted in Figure 1 and the following discusses the technical details.

3.2 Federated parameter decoupling and node augmentation

An intuitive manner to reach the above goal is to train the GNNs (e.g., GCN [48] or GraphSage [7]) using the basic FL algorithm FedAvg [10]. However, this naive strategy adopts the classification layers with

the same architecture for different clients to train and predict, akin to most PFL methods [22,35], which neglects the task heterogeneity among clients and thus could cause error-prone predictions, making it sub-optimal. To address this issue, our proposed `FedPANO` adopts parameter decoupling to decompose the local model into two distinct parts, i.e., a GNN and a classifier. Specifically, the GNN is induced using the FL framework to capture the general and shared knowledge across clients for learning node feature representation, while the classifier is specifically designed for each client and exclusively trained only locally to facilitate personalized predictions. Additionally, a generic classifier shared among clients is introduced to further encourage the GNN's grasp of general information, alleviating the bias introduced by the client-specific classifier. Nevertheless, building a well-performed client-specific classifier for clients with scarce graph nodes remains a demanding challenge. Given that, `FedPANO` introduces the node generator that leverages node correlation for each client to generate synthetic nodes, thereby enhancing the capabilities of the personalized classifier. Furthermore, the objective of the generator is designed to be universal across different clients, allowing it to undergo optimization through local training and further refinement through global aggregation. Following the general FL setup [10,28], `FedPANO` iteratively performs the local training and global aggregation steps. These two phases (one communication round) are presented below.

3.2.1 *Local training*

For each client c_i with graph data $G_i = \{\mathcal{V}_i, \mathcal{E}_i\}$, the local model first represents the query node $u \in \mathcal{V}_i$ based on its neighbors and itself using a K-layer GNN with parameters $\theta_i^g = \{\theta_i^{g_k}\}_{k=1}^K$, and then predicts the label of u according to the learnt representation using classifier with parameters θ_i^c, as follows:

$$\begin{aligned} \boldsymbol{h}_u^{k+1} &= \theta_i^{g_k}(\boldsymbol{h}_u^k || \mathrm{Agg}(\{\boldsymbol{h}_v^k, \forall v \in \mathcal{N}_{G_i}(u)\})), \\ \widetilde{\boldsymbol{y}}_u &= \theta_i^c(\boldsymbol{h}_u^{K+1}), \end{aligned} \tag{1}$$

where $\mathcal{N}_{G_i}(u)$ denotes the set of u's neighbors in graph G_i, $\mathrm{Agg}(*)$ is the node aggregator (e.g., mean pooling), $||$ is the concatenation operation, $\boldsymbol{h}_u^{k+1}$ is the representation feature obtained after $\theta_i^{g_k}$ and $\boldsymbol{h}_u^1$ is initialized as $\boldsymbol{x}_u$, $\widetilde{\boldsymbol{y}}_u$ indicates the predicted label for u. Thus the corresponding classification loss for c_i is computed as follows:

$$\mathcal{L}_i^c = \frac{1}{|\mathcal{V}_i|} \sum_{u \in \mathcal{V}_i} \mathrm{CE}(\widetilde{\boldsymbol{y}}_u, \boldsymbol{y}_u), \tag{2}$$

where $\mathrm{CE}(*)$ denotes the cross-entropy function. In this way, the parameters of local model $\theta_i = \{\theta_i^g, \theta_i^c\}$ can be optimized via the computed loss $\mathcal{L}_i^c$.

Parameter decoupling. The target label space $\{\mathcal{Y}_i\}_{i=1}^n$ for n clients is generally different ($\mathcal{Y}_i \neq \mathcal{Y}_j$) due to various preferences or usages, provoking the task heterogeneity. To tackle this issue, `FedPANO` first adopts the parameter decoupling to divide the local model θ_i into two parts as $\theta_i = \{\theta_i^g, \theta_i^c\}$, where θ_i^g and θ_i^c denote the GNN and classifier respectively, and then designs classifiers specific to each client. The targets of these two parts are defined as follows:

$$\underbrace{\theta_i^g : \mathcal{X} \to \mathcal{H}}_{\text{GNN module}} \quad \underbrace{\theta_i^c : \mathcal{H} \to \mathcal{Y}_i}_{\text{Classifier module}}, \tag{3}$$

where $\mathcal{X}$ is the original feature space of each client, and $\mathcal{H}$ denotes the new feature space learned by the GNN module, which is also shared across clients.

Regarding the GNN, since its purpose remains consistent for each client with the aim of capturing the shared and general knowledge across clients for representing node features, it can not only be optimized by local training but also be further advanced via global aggregation. However, existing decoupling methods (FedPer [24] and FedRep [25]) only adopt the feedback from the client-specific and unshared classifier θ_i^c to optimize θ_i^g, which would inevitably make θ_i^g overly impacted by the personalized knowledge, introducing the bias and causing the parameter divergence of GNN among clients. To alleviate this problem, different from FedPer and FedRep, `FedPANO` additionally employs a shared classifier parameterized by θ_i^s to further encourage GNN to model the general knowledge. Specifically, the target of θ_i^s is $\mathcal{H} \to \mathcal{Y}$, where $\mathcal{Y}$ represents the comprehensive label space across clients, and its prediction is defined as

$$\widetilde{\overline{\boldsymbol{y}}}_u = \theta_i^s(\boldsymbol{h}_u^{K+1}). \tag{4}$$

Then the corresponding loss $\mathcal{L}_i^s$ for training $\{\theta_i^g, \theta_i^s\}$ is calculated as follows:

$$\mathcal{L}_i^s = \frac{1}{|\mathcal{V}_i|} \sum_{u \in \mathcal{V}_i} \mathrm{CE}(\widetilde{\boldsymbol{y}}_u, \overline{\boldsymbol{y}}_u), \tag{5}$$

where $\overline{\boldsymbol{y}}_u$ denotes the converted label vector from $\mathcal{Y}_i$ to $\mathcal{Y}$ to match the output dimension of θ_i^s. Since the aim of θ_i^s is shared across clients, it not only encourages GNN to learn general knowledge but also boosts the global aggregation.

As for the client-specific classifier θ_i^c, its goal $\mathcal{H} \to \mathcal{Y}_i$ varies across clients due to task heterogeneity, thus it is optimized through the local data only to model the personalized task. However, scarce local training graph data for inducing the client-specific classifier should be further handled.

Node augmentation. To deal with this challenge, we introduce a node augmentation module using the node correlation to synthesize nodes for boosting the supervised information. As shown in Figure 1, the node augmentation module for each client c_i consists of a generator ϕ_i. The following describes its designs and local training strategy in detail.

The generator ϕ_i is designed to be a generative model that produces synthetic nodes with the same label for each real node. To enrich the characteristics of the few-shot training nodes, the generator should not only consider the real node itself but also its connected nodes (i.e., neighbors). Moreover, the generator is further equipped with a Gaussian noise generator $N(0,1)$ that outputs the corresponding noise vector to enhance the diversity of generated nodes. Mathematically, for each real node $u \in \mathcal{V}_i$, ϕ_i synthesize nodes with the same label to u using itself, its neighbors, and noise as follows:

$$\boldsymbol{h}_{\widehat{u}} = \phi_i(\boldsymbol{h}_u || \mathrm{Agg}(\{\boldsymbol{h}_v, \forall v \in \mathcal{N}_{G_i}(u)\}) + N(0,1)), \tag{6}$$

where $\boldsymbol{h}_{\widehat{u}}$ is the synthetic node feature vector for u, $\boldsymbol{h}_u/\boldsymbol{h}_v$ denotes the learnt representation $\boldsymbol{h}_u^{K+1}/\boldsymbol{h}_v^{K+1}$ using (1) for simplicity, and $||$ is the concatenation operation. The generated nodes $\boldsymbol{h}_{\widehat{u}}$ are assigned with the same labels as the real node u, because they are generated based on u and its neighbors, ensuring that the generated nodes accurately reflect the class properties of u. Noth that Eq. (6) can be repeated m times for a certain real node to enlarge training data, and we denote $\widehat{\mathcal{V}}_i$ as the synthetic node set of c_i.

For the local training of generator ϕ_i, `FedPANO` aims to maximize its capability to produce the synthetic node $\widehat{u}$ with the same label as the corresponding real node u. This ensures that the generated node $\widehat{u}$ is similar to the real node u. Thus the loss for training the generator can be computed as

$$\mathcal{L}_i^g = \frac{1}{|\mathcal{V}_i|} \sum_{u \in \mathcal{V}_i} \frac{1}{m} \sum_{j=1}^{m} d(\boldsymbol{h}_u, \boldsymbol{h}_{\widehat{u}_j}), \tag{7}$$

where $\boldsymbol{h}_{\widehat{u}_j}$ denotes the j-th synthetic node for u, $d(\boldsymbol{h}_u, \boldsymbol{h}_{\widehat{u}_j})$ estimates the Euclidean distance between $\boldsymbol{h}_u$ and $\boldsymbol{h}_{\widehat{u}_j}$, and $\mathcal{L}_i^g$ is the loss of c_i in training generator ϕ^g.

The well-performance generator ϕ_i can hardly be induced with the limited training local data. Thus it should be further enhanced by collaborative training through FL. To achieve this, the target of the node augmentation module is designed to be consistent across clients as (6) and (7), which can be alternatively expressed as $\phi_i : \mathcal{H} \times \mathcal{H} \to \mathcal{H}$. ϕ_i is conceived to generate synthetic nodes using the real nodes and their neighbors, so its aim is compatible among clients.

Classifier strengthen. With the synthetic node set $\widehat{\mathcal{V}}_i$ obtained via the node generator, `FedPANO` further elevates the client-specific classifier θ_i^c for each client c_i as

$$\begin{aligned} \widetilde{\boldsymbol{y}}_{\widehat{u}} &= \theta_i^c(\boldsymbol{h}_{\widehat{u}}), \\ \mathcal{L}_i^{\widehat{c}} &= \frac{1}{|\widehat{\mathcal{V}}_i|} \sum_{\widehat{u} \in \widehat{\mathcal{V}}_i} \mathrm{CE}(\widetilde{\boldsymbol{y}}_{\widehat{u}}, \boldsymbol{y}_{\widehat{u}}), \end{aligned} \tag{8}$$

where $\widetilde{\boldsymbol{y}}_{\widehat{u}}$ is prediction results of θ_i^c for $\widehat{u}$, and $\mathcal{L}_i^{\widehat{c}}$ denotes loss of c_i in training classifier θ_i^c with $\widehat{\mathcal{V}}_i$.

Overall local training. The local model for each client c_i consists of four components, i.e., a GNN θ_i^g, a shared classifier θ_i^s, a client-specific classifier θ_i^c and a node generator ϕ_i. The θ_i^g, θ_i^s and θ_i^c are first trained via $\mathcal{L}_i^s$ and $\mathcal{L}_i^c$ computed in (5) and (2), respectively. Then `FedPANO` optimizes ϕ_i with $\mathcal{L}_i^g$ calculated in (7). With the help of synthetic nodes, `FedPANO` further improves θ_i^c using $\mathcal{L}_i^{\widehat{c}}$ estimated in (8). Thus, the overall local training of client c_i is obtained by optimizing the following:

$$\mathcal{L}_i = \mathcal{L}_i^s + \mathcal{L}_i^c + \mathcal{L}_i^g + \mathcal{L}_i^{\widehat{c}}. \tag{9}$$

Algorithm 1 FedPANO.

Input: n clients, where each client c_i carries local data G_i, local test node u_i on client c_i.
Parameter: $m = 5$ (number of synthetic nodes for each real node); $T = 500$ (number of communication rounds).
Output: Trained personalized models $\{(\theta_i, \phi_i)\}_{i=1}^n$.

1: Server initializes the GNN, shared classifier, and node generator of clients as θ_*^g, θ_*^s and ϕ_*;
2: Clients initialize their client-specific classifier as θ_i^c;
3: **for** $t = 1 \to T$ **do**
4: **for** all clients $i = 1 \to n$ **in parallel do**
5: Optimize GNN θ_i^g, shared classifier θ_i^s, client-specific classifier θ_i^c via (2) and (5);
6: Update generator ϕ_i through (7);
7: Further boost classifier θ_i^c with (8);
8: Send models $(\theta_i^g, \theta_i^s, \phi_i)$ to server;
9: **end for**
10: **for** server **do**
11: Receive $\{(\theta_i^g, \theta_i^s, \phi_i)\}_{i=1}^n$ from n clients;
12: Perform model aggregation via (10);
13: Broadcast models $(\theta_*^g, \theta_*^s, \phi_*)$ to each c_i;
14: **end for**
15: **end for**
16: Client c_i classifies u_i using learned θ_i^g and θ_i^c.

3.2.2 *Model aggregation*

The local training of the node augmentation module allows it to generate synthetic nodes that are beneficial for boosting the client-specific classifier. However, limited supervised information remains a challenge for training the GNN, shared classifier, and node augmentation module, consequently impairing the client-specific classifier's performance. To address this problem, since clients share the same objectives for GNN, generic classifier, and node generator, FedPANO further improves them collaboratively with model aggregation (e.g., FedAvg) as follows:

$$\begin{aligned} \theta_*^{g/s} &= \sum_{i=1}^{n} \frac{|\mathcal{V}_i|}{\sum_{j=1}^{n} |\mathcal{V}_j|} \theta_i^{g/s}, \\ \phi_* &= \sum_{i=1}^{n} \frac{|\mathcal{V}_i|}{\sum_{j=1}^{n} |\mathcal{V}_j|} \phi_i, \end{aligned} \tag{10}$$

where $\theta_*^{g/s}$ and ϕ_* denote the aggregated GNN and shared classifier, and node generator, respectively. They will be broadcast to participating clients for the next communication round. Note that other model aggregation methods [37,49] can also be smoothly merged with our FedPANO. Regarding the client-specific classifier, it is optimized locally only to model the personalized tasks, alleviating the problem of task heterogeneity. FedPANO iterates between local training and global aggregation until reaching the communication round T. Algorithm 1 summarizes the two phases of FedPANO, local training (steps 5–8), and global aggregation (steps 11–13).

For the privacy concerns of FedPANO, following the general FL setting, it only shares the model parameters of clients, in which the data privacy is not maliciously compromised. Furthermore, many strategies can be applied to safeguard the privacy of client models in FL, such as differential privacy [50,51], secure aggregation [52], and homomorphic encryption (HE) [53], which can be readily applied into FedPANO. As an example of utilizing HE, during the training phase, clients securely synchronize an HE key pair, encrypt their local models using the public key, and transmit the ciphertexts to the central server. Then the server aggregates these encrypted models and sends the results back to the clients, who subsequently decrypt and update their models accordingly [54]. This process ensures data privacy throughout transmission and aggregation. However, it is important to note that this aspect does not form the primary research focus of this study.

3.3 Communication and computational cost

During the communication between each client and server, our FedPANO involves three model parameters: a GNN, a shared classifier, and a generator. Compared with FedAvg, FedPANO sends an additional generator to each client and sends it back to the server; as such, the total and the extra communication cost in each round is $\mathrm{CL}(\theta_i^g) + \mathrm{CL}(\theta_i^s) + \mathrm{CL}(\phi_i)$ and $\mathrm{CL}(\phi_i)$, respectively, where $\mathrm{CL}(*)$ denotes the communication load of $*$. Since the generator is configured as a two-layer fully connected layer, as detailed in Section 4, the additional communication cost is much less than the main overhead.

Table 1 Statistics of four datasets.

Dataset	#Graphs	#Nodes	#Edges	#Features	#Classes
DBLP	1	40672	288270	7202	137
Cora-full	1	19793	65311	8710	70
Amazon-C	1	24919	91680	9034	77
Fold-PPI	144	274606	3666563	512	29

Compared with other FL methods, the primary extra computational cost of `FedPANO` is to compute $\mathcal{L}_i^g$ and $\mathcal{L}_i^{\hat{c}}$ for training generator (a two-layer fully connected layer) and client-specific classifier (one fully connected layer), respectively, in which the number of generated samples plays a crucial role. The extra computational cost is insignificant because of simple model architectures, and we empirically observe that effective performance can be achieved with m (number of generated nodes for each real node) $\leqslant 5$.

4 Experiment

4.1 Experimental setup

Datasets. We conduct experiments on four public datasets (DBLP [55], Cora-full [56], Amazon-C [57], and Fold-PPI [47]) to prove the effectiveness of our proposed method.

The detailed statistics of these datasets are provided in Table 1. To simulate the decentralized data distributions, for the standalone-graph DBLP, Cora-full and Amazon-C, following [22], we construct distributed subgraphs by dividing the dataset into a certain number of participants using METIS algorithm [58], such that each FL participant has a subgraph that is part of an original graph. As for the multiple-graph Fold-PPI, each participating client is randomly assigned to a distinct network, and 1/3 categories in each network are randomly chosen for training and testing to emulate task heterogeneity. Then for each client, we adopt categories with more than 25 samples (combined for training and testing) as the target labels. Within each category, K nodes and another 15 nodes are randomly selected for training and testing, respectively.

Baselines. To perform a comprehensive comparison, we compare `FedPANO` against two typical global FL methods (FedAvg [10] and FedProx [28]), two representative PFL methods (FedPer [24] and FedRep [25]), two PFL methods on decentralized graphs (FedSage+ [21], FED-PUB [22]), and two recent FL methods on few-shot data (pFedFSL [29] and F^{2}L [30]). In addition, we also provide the results of Local as a naive baseline, in which models are trained solely on clients. Each compared method is configured with the suggested parameters in the corresponding literature. For each method, we report the mean and standard deviation of the accuracy of 5 independent runs, where the average accuracy of all clients is recorded in each run.

Implementation details. To perform a fair comparison, all compared methods utilize the following configurations: the optimizer SGD with learning rates of 0.1/0.5/0.1/0.01 for datasets DBLP/Cora-full/Amazon-C/Fold-PPI, a two-layer GraphSage [7] with the mean aggregator as the base GNN and a one-layer fully connected layer as the classifier, 5 epochs in each local update, 5 participating clients in each communication round, and 500 communication rounds for FL methods. As to the node generator in our `FedPANO`, it is configured as the two-layer fully connected layer with ReLU and Tanh non-linearity, respectively, and the number of synthetic nodes for each real node is set to $m = 5$.

4.2 Experimental results

We conduct experiments on four datasets under the scenarios with 10 or 30 clients as $K \in \{3, 5, 10\}$. The results of each compared method are presented in Table 2. From this table, we have the following observations.

(i) `FedPANO` frequently outperforms other methods across different settings, which validates the effectiveness of `FedPANO` in the personalized federated few-shot node classification landscape. As the number of clients steps from 10 to 30 under a fixed K, nearly every method displays improved performance. This improvement can be attributed to the reduction in the average number of classes per client for DBLP/Cora-full/Amazon-C (from 40.7/25.2/26.7 to 19.8/10.6/12.7), resulting in decreased task complexity due to graph partition, while more distributed graph data can be utilized to collaboratively learn for Fold-PPI. On the other hand, as K increases under a fixed number of clients, each method exhibits

Table 2 Classification accuracy (mean±std) of comparison methods. K: the number of training samples per class for each client. The best performance in each setting is bold-faced. ○/● indicates that `FedPANO` is statistically worse/better than the compared method by student pairwise t-test at 95% confident level.

	10 clients			30 clients		
	$K=3$	$K=5$	$K=10$	$K=3$	$K=5$	$K=10$
			DBLP			
Local	19.10±0.72●	20.90±0.62●	25.54±0.66●	20.78±0.35●	23.49±0.63●	26.86±0.56●
FedAvg	20.55±0.76●	22.14±0.51●	26.05±0.29●	21.95±0.46●	24.20±0.54●	27.65±0.61
FedProx	20.75±0.38●	22.01±0.51●	26.17±0.33●	21.75±0.38●	24.54±0.58●	28.09±0.31
FedPer	20.10±0.54●	21.66±0.37●	27.78±0.65	21.26±0.57●	25.34±0.38●	28.12±0.47
FedRep	20.01±0.58●	21.74±0.39●	26.55±0.36	21.14±0.48●	25.38±0.46●	28.11±0.48
pFedFSL	20.86±0.63●	22.56±0.53●	26.90±0.35	22.36±0.45●	24.86±0.63●	26.82±0.69●
F^2L	20.73±0.52●	22.71±0.63●	26.85±0.43	22.17±0.24●	25.56±0.59●	27.02±0.39●
FedSage+	20.49±0.57●	22.56±0.37●	27.63±0.48	22.57±0.28●	24.97±0.52●	28.43±0.56
FED-PUB	20.62±0.64●	22.23±0.64●	**28.13±0.35**○	22.37±0.57●	25.09±0.44●	**28.94±0.40**○
`FedPANO`	**21.90±0.58**	**23.95±0.47**	27.09±0.44	**23.92±0.33**	**26.45±0.39**	28.05±0.53
			Cora-full			
Local	42.42±0.98●	44.69±0.17●	48.09±1.27●	50.65±1.30●	55.05±0.85●	60.81±1.62●
FedAvg	46.16±0.50●	52.72±1.32●	56.97±1.18●	54.54±0.89●	59.28±1.11●	64.90±0.80●
FedProx	46.11±0.47●	52.82±1.27●	56.91±1.20●	54.33±0.74●	59.16±1.34●	64.65±0.73●
FedPer	43.25±0.66●	50.58±0.92●	58.36±0.81●	53.80±0.73●	58.73±0.74●	65.28±0.67●
FedRep	43.68±0.71●	50.81±0.94●	58.34±0.80●	52.51±0.78●	58.43±1.00●	65.53±0.67●
pFedFSL	48.57±0.56●	53.06±0.42●	59.06±0.42●	55.80±0.54●	60.59±0.51●	65.15±0.89●
F^2L	47.97±0.38●	53.78±0.89●	59.18±0.53●	55.63±0.66●	60.18±0.61●	65.65±0.73●
FedSage+	48.79±0.46●	54.13±0.61●	60.31±0.57	57.06±0.64●	61.01±0.36●	65.87±0.40●
FED-PUB	49.19±0.53●	54.58±0.92	**61.31±0.33**	56.46±0.39●	61.48±0.79●	66.07±0.54●
`FedPANO`	**51.31±0.80**	**56.19±1.06**	60.52±0.67	**58.67±0.66**	**63.86±0.64**	**67.50±0.69**
			Amazon-C			
Local	43.70±0.95●	46.35±1.15●	50.27±0.73●	51.94±0.65●	56.06±0.46●	60.06±0.24●
FedAvg	47.62±0.86●	54.74±1.04●	57.61±0.84●	55.24±0.98●	58.33±0.52●	63.08±0.64●
FedProx	47.69±0.73●	54.89±0.95●	58.08±0.85●	54.45±1.03●	59.38±0.55●	63.32±0.56●
FedPer	44.65±0.44●	52.68±1.15●	59.84±0.37●	53.34±0.71●	58.40±0.56●	64.06±0.96●
FedRep	44.99±0.46●	52.95±1.23●	60.02±0.45●	52.98±0.92●	59.04±0.55●	64.32±0.93●
pFedFSL	48.40±0.61●	53.40±0.67●	59.87±0.70●	56.25±0.24●	60.33±0.55●	64.58±0.92●
F^2L	48.75±0.84●	53.93±0.50●	59.66±0.67●	56.65±0.72●	60.67±0.82●	64.98±0.90●
FedSage+	49.53±0.70●	55.06±0.39●	59.95±0.80●	56.64±0.64●	61.07±0.71●	65.11±0.59●
FED-PUB	50.03±0.81●	55.55±0.57●	60.33±0.89●	55.64±1.23●	60.57±0.45●	65.02±0.53●
`FedPANO`	**52.74±0.63**	**57.47±0.55**	**61.72±0.68**	**59.36±0.53**	**62.28±0.67**	**66.23±0.32**
			Fold-PPI			
Local	42.32±0.36●	45.36±0.59●	48.55±0.87●	43.47±0.62●	46.57±0.83●	47.95±0.60●
FedAvg	53.33±0.80●	61.00±0.43●	69.70±0.83●	65.53±0.82●	70.44±1.13●	73.95±0.96●
FedProx	53.21±0.59●	62.67±0.90●	68.74±0.61●	65.63±0.74●	71.24±1.01●	74.22±0.93●
FedPer	51.55±0.57●	61.59±0.68●	70.55±0.82●	64.35±0.46●	70.62±1.33●	75.81±1.16●
FedRep	51.32±0.72●	63.84±0.72●	71.53±0.92●	63.36±0.65●	70.42±0.90●	75.87±0.79●
pFedFSL	56.57±0.85●	64.52±0.62●	72.82±0.43●	68.81±0.59●	72.59±1.06●	73.06±0.73●
F^2L	56.01±0.70●	64.72±0.56●	72.18±0.54●	69.26±0.79●	73.09±0.89●	73.70±0.87●
FedSage+	55.05±0.46●	63.53±0.61●	71.88±0.50●	66.93±0.63●	74.10±0.67●	74.60±0.75●
FED-PUB	55.79±0.34●	63.12±0.77●	72.18±0.61●	67.42±0.77●	73.95±0.49●	74.29±0.84●
`FedPANO`	**59.03±0.69**	**68.05±0.63**	**74.63±0.66**	**75.33±0.61**	**76.92±0.64**	**79.10±0.53**

an increasing accuracy owing to the availability of more training nodes. The performance gap between `FedPANO` and the compared methods is more prominent when K is small, providing further confirmation of `FedPANO`'s superiority in effectively addressing scenarios with limited training nodes.

(ii) Global FL vs. PFL. The local method is surpassed by all FL methods, thereby showcasing the necessity and effectiveness of FL. Within scenarios where the value of K is small, global FL methods (FedAvg and FedProx) often outperform PFL methods (FedPer and FedRep), owing to the personalized layers being difficult to train with extremely limited training data. Conversely, in cases where K is large,

Table 3 Performance of FedPANO and its five degenerated variants on four datasets under scenarios with 10 or 30 clients and $K = 3$. The best performance in each setting is bold-faced. ○/● indicates that FedPANO is statistically worse/better than variant by student pairwise t-test at 95% confident level.

	FedPANO-nT	FedPANO-nS	FedPANO-nN	FedPANO-nF	FedPANO-nG	FedPANO
			10 clients			
DBLP	20.77±0.43●	21.00±0.34●	20.59±0.38●	21.06±0.51	20.23±0.37●	**21.90±0.58**
Cora-full	48.56±0.61●	49.69±0.73●	47.43±0.61●	49.45±0.89●	49.12±0.57●	**51.31±0.80**
Amazon-C	49.23±0.74●	50.49±0.60●	48.96±0.89●	50.16±0.99●	50.66±0.79●	**52.74±0.63**
Fold-PPI	53.83±0.83●	54.21±0.54●	54.75±0.64●	57.62±0.68●	53.05±0.75●	**59.03±0.69**
			30 clients			
DBLP	22.57±0.49●	22.53±0.54●	22.08±0.66●	22.43±0.42●	22.86±0.59●	**23.92±0.33**
Cora-full	55.69±0.56●	54.21±0.97●	55.25±0.70●	56.10±0.70●	55.67±0.69●	**58.67±0.66**
Amazon-C	56.52±0.61●	55.64±0.71●	56.89±0.46●	57.70±0.75●	57.81±0.68●	**59.36±0.53**
Fold-PPI	70.80±0.11●	71.71±0.55●	72.38±0.53●	73.30±0.62●	73.19±0.64●	**75.33±0.61**

global FL methods trail behind the PFL methods. This is attributed to that a single global model cannot fit non-i.i.d. local data well. These methods generally lose to FedSage+ and FED-PUB, which consider node dependency and are designed for decentralized graph scenarios. However, FedSage+ and FED-PUB are still outperformed by FedPANO, since they are bounded with limited training nodes. In addition, they still adopt uniform classification layers for different clients and overlook task heterogeneity. In contrast, our FedPANO decouples the local model into a GNN and a client-specific classifier for task heterogeneity, and it incorporates an extra classifier shared among clients to incentivize GNN to capture the general knowledge. Furthermore, FedPANO handles the scarce training nodes by the meticulously designed node generator.

(iii) FedPANO vs. pFedFSL and F^{2}L. Although pFedFSL and F^{2}L address the few-shot data problems within the FL framework by the learning of embedding network and meta-knowledge, respectively, they still fall short when compared to FedPANO. This is because they overlook the node relationships when refining embedding spaces and acquiring meta-knowledge. It also suggests the effectiveness of FedPANO utilizing the node interactions to enhance the node generator for boosting supervised information.

4.3 Ablation study

We conduct an ablation study to gain deeper insights into the contributing factors of FedPANO. For this purpose, we introduce five variants of FedPANO: FedPANO-nT, FedPANO-nS, FedPANO-nN, FedPANO-nF, and FedPANO-nG. Specifically, FedPANO-nT excludes the use of client-specific classifier and instead employs a uniform classifier for each client during training and inference; FedPANO-nS omits the shared classifier that encourages GNN to capture general knowledge among clients; FedPANO-nN trains local models with only real nodes, excluding the node generator; FedPANO-nF involves training the node generator locally only, without collaborative training via FL; FedPANO-nG performs node augmentation by the addition of node representation and small random Gaussian noise (produced by $N(0, 0.1)$), without using the generator. Table 3 presents the results of five variants and FedPANO under scenarios with 10 or 30 clients and $K = 3$. From this table, we can observe that:

(i) The discernible performance gap between FedPANO and FedPANO-nT underscores the necessity and effectiveness of employing the client-specific classifier for each client to tackle task heterogeneity.

(ii) The presence of the shared classifier among clients helps GNN to capture general knowledge and enhance model performance. This assertion is substantiated by the notable margin between FedPANO and FedPANO-nS.

(iii) The fact that FedPANO-nF outperforms FedPANO-nN shows the potency of the node generator in amplifying supervised information. Nevertheless, it still falls short when compared to FedPANO, which confirms the effectiveness of the meticulous strategy for designing node generator so that it can be further improved through FL.

(iv) The performance of FedPANO-nG surpassing FedPANO-nN again underscores the effectiveness of node augmentation for boosting supervised information. However, it does not reach the performance level of FedPANO, confirming the quality of generated samples by our designed node generator, which considers dependencies between nodes to enrich the characteristics of training nodes.

Table 4 Accuracy of FedPANO with different GNNs on four datasets under scenarios with 10 clients and $K = 3$.

	DBLP	Cora-full	Amazon-C	Fold-PPI
GCN	20.09±0.49	50.53±0.63	51.83±0.71	57.68±0.98
GAT	22.35±0.67	53.13±0.58	53.91±0.50	58.19±0.76
GraphSage	21.90±0.58	51.31±0.80	52.74±0.63	59.03±0.69

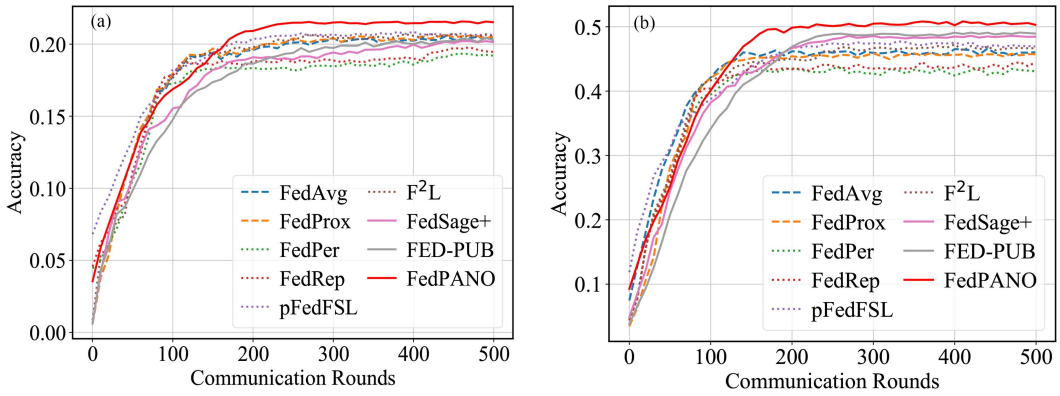

Figure 2 (Color online) Accuracy of compared FL methods vs. communication rounds under the scenarios with 10 clients and $K = 3$. (a) DBLP; (b) Cora-full.

4.4 Impact of different GNNs

To investigate the influence of different GNNs on our FedPANO, we adopt three commonly used GNN architectures as the underlying backbone to perform experiments. The performance of FedPANO on four datasets under scenarios with 10 clients and $K = 3$ using different GNNs, including graph convolutional network (GCN) [48], graph attention network (GAT) [59], and GraphSage [7], are detailed in Table 4. From the table, it is evident that FedPANO consistently delivers robust performance across different GNNs. This consistency highlights the compatibility of FedPANO with different GNNs, demonstrating the robustness and adaptability of FedPANO.

4.5 Convergence analysis

Figure 2 depicts the test accuracy of each compared FL method during the training process on DBLP and Cora-full under the 10 clients scenario with $K = 3$. From these figures, it can be observed that FedPANO achieves its peak accuracy within approximately 200 communication rounds. This observation signifies a convergence rate comparable to other methods, yet with exceptional classification accuracy. These results again underscore the practicality and superiority of FedPANO in the face of task heterogeneity and scarce training nodes.

4.6 Impact of the number of client

In this experiment, we assess the impact of a wider range of client numbers. Figure 3 shows the accuracy of each compared FL method on DBLP and Cora-full as the number of clients n increases from 10 to 100 with $K = 3$. Each method has an improved performance as n increases, due to the reduction in the average number of labels per client (e.g., 40.7/25.2 to 8.6/3.8 for DBLP/Cora-full as n raises from 10 to 100) resulting from graph partition, which decreases the task complexity. Moreover, FedPANO consistently outperforms the others across different n, which again demonstrates its effectiveness.

4.7 Impact of shared classifier

FedPANO employs a shared and universal classifier to mitigate the parameter discrepancies in GNNs across clients that are induced by the customized classifier. This classifier aids the GNNs in capturing collective and overarching insights shared among clients. To verify this proposition, we investigate the average distance of GNNs' parameters between clients during the training process, evaluated using Euclidean distance, i.e., Distance $= \frac{2}{n\times(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \|\theta_i^g - \theta_j^g\|_2$, where n represents the number of clients and

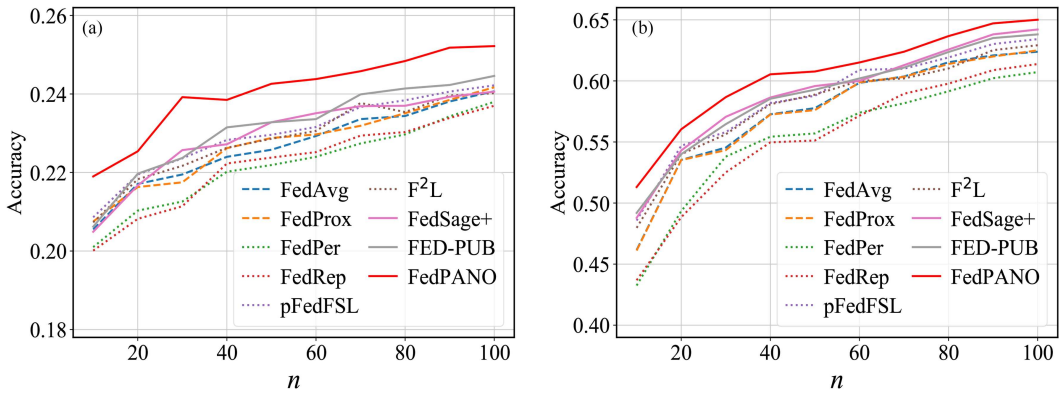

Figure 3 (Color online) Accuracy of compared FL methods vs. number of clients n with $K = 3$. (a) DBLP; (b) Cora-full.

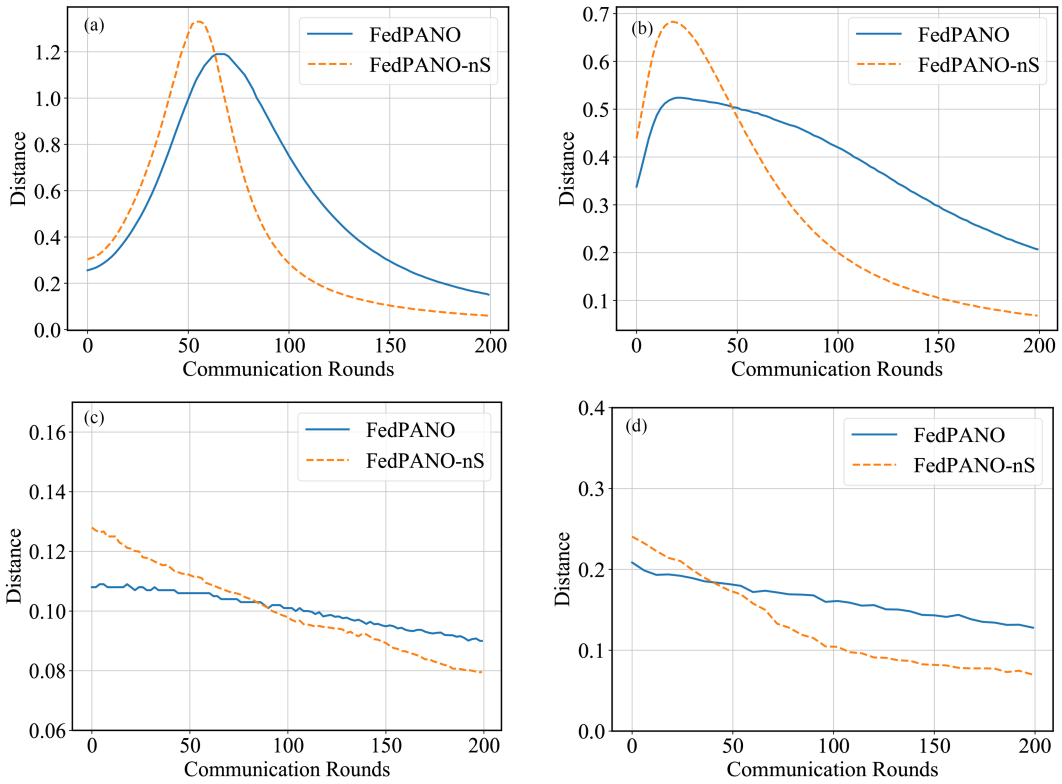

Figure 4 (Color online) Distance of model parameters (parameter discrepancy) vs. communication rounds on four datasets under scenarios with 10 clients and $K = 3$. `FedPANO` demonstrates a notable advantage in terms of lower parameter discrepancies compared to `FedPANO`-nS during the initial rounds. (a) DBLP; (b) Cora-full; (c) Amazon-C; (d) Fold-PPI.

larger distances indicate substantial parameter divergence. The distance during the first 200 communication rounds of `FedPANO` and `FedPANO`-nS (omitting the shared classifier) on four datasets under the scenario with 10 clients and $K = 3$ are presented in Figure 4. As illustrated in Figure 4, `FedPANO` exhibits a noticeable advantage in terms of lower parameter discrepancies compared to `FedPANO`-nS during the initial rounds. This observation demonstrates the advantages of the shared classifier in mitigating the divergence in GNN parameters across clients. However, as the round further iterates, the parameter discrepancies of `FedPANO` begin to surpass those of `FedPANO`-nS. This phenomenon can be attributed to the local models of `FedPANO`-nS gradually approaching convergence, prematurely resulting in diminished client differences, which compromises the model's accuracy.

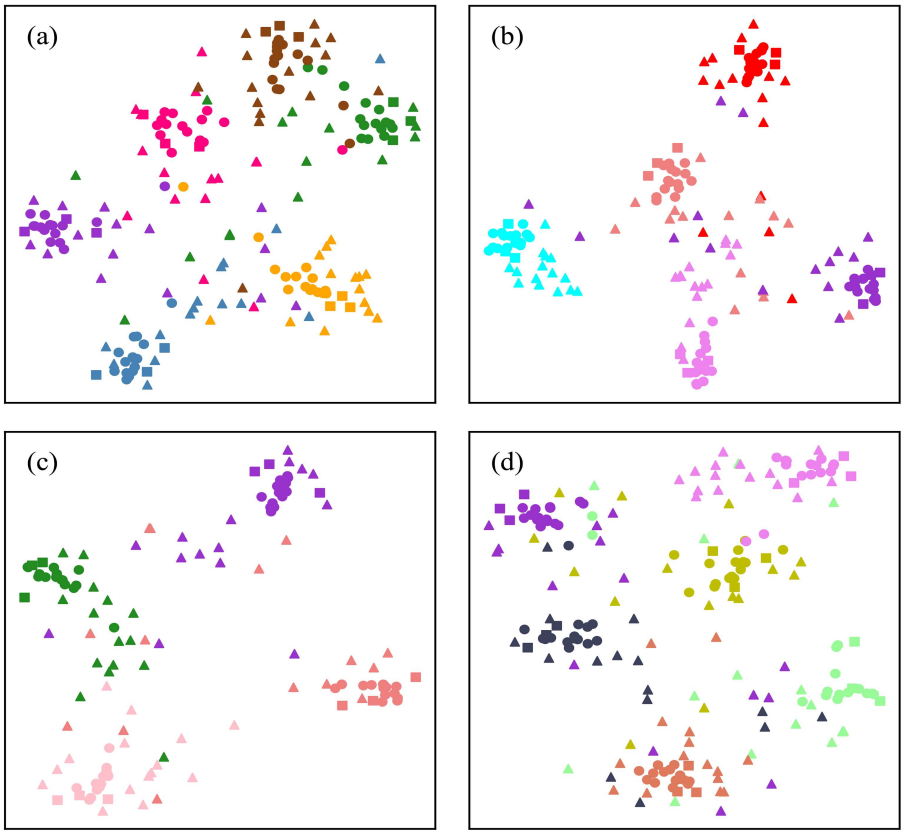

Figure 5 (Color online) Visualization of synthetic nodes (circle), training nodes (square), and test nodes (triangle) on Fold-PPI owned by four clients under scenarios with 10 clients and $K = 3$, and diverse colors indicate different classes. The synthetic nodes typically cluster near the real training nodes, showing slight dissimilarities with them. (a) Client A; (b) client B; (c) client C; (d) client D.

4.8 Visualization of synthetic nodes

`FedPANO` learns a node generator for each client to synthesize nodes. To further assess its effectiveness, we visualize these synthetic nodes (represented as circles) alongside the original training nodes (depicted as squares) and test nodes (shown as triangles) for four certain clients on Fold-PPI under the scenario involving 10 clients and $K = 3$ using t-SNE [60] in Figure 5, where nodes are situated in the new feature space $\mathcal{H}$ learned via the GNNs and diverse colors denoting distinct labels. As illustrated in Figure 5, the synthetic nodes generally cluster in proximity to the real training nodes, exhibiting slight dissimilarities with them. This process enriches the traits and diversity of the few-shot training nodes, subsequently enhancing the performance of downstream classifiers. Consequently, this process results in more accurate predictions for the testing nodes, affirming the effectiveness of the synthesized nodes generated by `FedPANO`.

5 Conclusion and future work

This paper addresses a practical and intricate problem within the realm of FL, where clients have heterogeneous tasks but with few-shot training nodes. Our introduced `FedPANO` separates the local model into a GNN and a classifier, enabling the accommodation of client-specific task variations. It trains GNN within FL to capture shared insights across clients, additionally facilitated by a shared and universal classifier among clients, meanwhile managing a specific-designed classifier on each client for task heterogeneity. Furthermore, `FedPANO` carefully designs a node generator for each client with localized and collaborative training strategies to break the bottleneck of limited training nodes. Extensive experiments confirm the superiority of `FedPANO` for dealing with task heterogeneity and scarce training graph nodes.

In the future, given the increasing prevalence of unlabeled nodes in real-world graph data, we plan to

extend our solution to incorporate and benefit from this larger pool of unlabeled data.

Acknowledgements This work was partially supported by National Key Research and Development Program of China (Grant No. 2023YFF0725500), National Natural Science Foundation of China (Grant No. 62031003), Taishan Scholar Project Special Funding, and the Xiaomi Young Talents Program.

References

1 Zhang Z, Cui P, Zhu W. Deep learning on graphs: a survey. IEEE Trans Knowl Data Eng, 2022, 34: 249–270
2 Fu S C, Peng Q M, He Y, et al. Unsupervised multiplex graph diffusion networks with multi-level canonical correlation analysis for multiplex graph representation learning. Sci China Inf Sci, 2025, 68: 132101
3 Xia R, Zhang C, Zhang Y, et al. A novel graph oversampling framework for node classification in class-imbalanced graphs. Sci China Inf Sci, 2024, 67: 162101
4 Wen H Z, Ding J Y, Jin W, et al. Graph neural networks for multimodal single-cell data integration. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2022. 4153–4163
5 Zhang Z, Zhuang F Z, Zhu H S, et al. Relational graph neural network with hierarchical attention for knowledge graph completion. In: Proceedings of AAAI Conference on Artificial Intelligence, 2020. 9612–9619
6 Chen Y, Wu L, Zaki M J. Toward subgraph-guided knowledge graph question generation with graph neural networks. IEEE Trans Neural Netw Learn Syst, 2024, 35: 12706–12717
7 Hamilton W, Ying Z T, Leskovec J. Inductive representation learning on large graphs. In: Proceedings of Advances in Neural Information Processing Systems, 2017. 1024–1034
8 Song G, Li Y, Wang J, et al. Inferring explicit and implicit social ties simultaneously in mobile social networks. Sci China Inf Sci, 2020, 63: 149101
9 Gao Z H, Chen X M, Shao X D. Robust federated learning for edge-intelligent networks. Sci China Inf Sci, 2022, 65: 132306
10 McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data. In: Proceedings of International Conference on Artificial Intelligence and Statistics, 2017. 1273–1282
11 Kang X P, Yu G X, Wang J, et al. Incentive-boosted federated crowdsourcing. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2023. 6021–6029
12 Kang X P, Yu G X, Li Q Z, et al. Semi-asynchronous online federated crowdsourcing. In: Proceedings of IEEE International Conference on Data Engineering, 2024. 4180–4193
13 Yang Q, Liu Y, Chen T, et al. Federated machine learning: concept and applications. ACM Trans Intell Syst Technol, 2019, 10: 1–19
14 Chen C C, Zhou J, Zheng L F, et al. Vertically federated graph neural network for privacy-preserving node classification. In: Proceedings of the International Joint Conference on Artificial Intelligence, 2022. 1959–1965
15 Smith V, Chiang C K, Sanjabi M, et al. Federated multi-task learning. In: Proceedings of Advances in Neural Information Processing Systems, 2017. 4427–4437
16 Zhang Z, Zhang Y, Guo D, et al. Communication-efficient federated continual learning for distributed learning system with non-IID data. Sci China Inf Sci, 2023, 66: 122102
17 Liao X T, Chen C C, Liu W M, et al. Joint local relational augmentation and global Nash equilibrium for federated learning with non-IID data. In: Proceedings of the 31st ACM International Conference on Multimedia, 2023. 1536–1545
18 Fallah A, Mokhtari A, Ozdaglar A. Personalized federated learning with theoretical guarantees: a model-agnostic meta-learning approach. In: Proceedings of Advances in Neural Information Processing Systems, 2020. 3557–3568
19 Marfoq O, Neglia G, Vidal R, et al. Personalized federated learning through local memorization. In: Proceedings of International Conference on Machine Learning, 2022. 15070–15092
20 Wu J, Bao W X, Ainsworth E, et al. Personalized federated learning with parameter propagation. In: Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023. 2594–2605
21 Zhang K, Yang C, Li X X, et al. Subgraph federated learning with missing neighbor generation. In: Proceedings of Advances in Neural Information Processing Systems, 2021. 6671–6682
22 Baek J, Jeong W, Jin J, et al. Personalized subgraph federated learning. In: Proceedings of International Conference on Machine Learning, 2023. 1396–1415
23 Tan Y, Liu Y X, Long G D, et al. Federated learning on non-IID graphs via structural knowledge sharing. In: Proceedings of AAAI Conference on Artificial Intelligence, 2023. 9953–9961
24 Arivazhagan M G, Aggarwal V, Singh A K, et al. Federated learning with personalization layers. 2019. ArXiv:1912.00818
25 Collins L, Hassani H, Mokhtari A, et al. Exploiting shared representations for personalized federated learning. In: Proceedings of International Conference on Machine Learning, 2021. 2089–2099
26 Zhang C X, Ding K Z, Li J D, et al. Few-shot learning on graphs. In: Proceedings of International Joint Conferences on Artificial Intelligence, 2022. 5662–5669
27 Zhang L, Wang S, Liu J, et al. MuL-GRN: multi-level graph relation network for few-shot node classification. IEEE Trans Knowl Data Eng, 2023, 35: 6085–6098
28 Li T, Sahu A K, Zaheer M, et al. Federated optimization in heterogeneous networks. In: Proceedings of Machine Learning and Systems, 2020. 429–450
29 Zhao Y, Yu G, Wang J, et al. Personalized federated few-shot learning. IEEE Trans Neural Netw Learn Syst, 2024, 35: 2534–2544
30 Wang S, Fu X B, Ding K Z, et al. Federated few-shot learning. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023. 2374–2385
31 Tan A Z, Yu H, Cui L, et al. Towards personalized federated learning. IEEE Trans Neural Netw Learn Syst, 2023, 34: 9587–9603
32 Dinh C T, Tran N H, Nguyen T D. Personalized federated learning with Moreau envelopes. In: Proceedings of Advances in Neural Information Processing Systems, 2020. 21394–21405
33 Yang M, Wang X M, Zhu H B, et al. Federated learning with class imbalance reduction. In: Proceedings of European Signal Processing Conference, 2021. 2174–2178
34 Duan M, Liu D, Chen X, et al. Self-balancing federated learning with global imbalanced data in mobile systems. IEEE Trans Parallel Distrib Syst, 2020, 32: 59–71
35 Zhang M, Sapra K, Fidler S, et al. Personalized federated learning with first order model optimization. In: Proceedings of International Conference on Learning Representations, 2021. 1–12
36 Sattler F, Muller K R, Samek W. Clustered federated learning: model-agnostic distributed multitask optimization under privacy constraints. IEEE Trans Neural Netw Learn Syst, 2021, 32: 3710–3722
37 Ma X S, Zhang J, Guo S, et al. Layer-wised model aggregation for personalized federated learning. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2022. 10092–10101
38 Wang Y, Yao Q, Kwok J T, et al. Generalizing from a few examples: a survey on few-shot learning. ACM Comput Surv, 2021, 53: 1–34

39 Zhao Y F, Yu G X, Liu L, et al. Few-shot partial-label learning. In: Proceedings of International Joint Conference on Artificial Intelligence, 2021. 3448–3454
40 Zhao Y F, Yu G X, Liu L, et al. Few-shot partial multi-label learning. In: Proceedings of IEEE International Conference on Data Mining, 2021. 926–935
41 Liu Z M, Nguyen T K, Fang Y. Tail-GNN: tail-node graph neural networks. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2021. 1109–1119
42 Zhang Y M, Qian Y Y, Ye Y F, et al. Adapting distilled knowledge for few-shot relation reasoning over knowledge graphs. In: Proceedings of SIAM International Conference on Data Mining, 2022. 666–674
43 Wang S, Huang X, Chen C, et al. Reform: error-aware few-shot knowledge graph completion. In: Proceedings of ACM International Conference on Information & Knowledge Management, 2021. 1979–1988
44 Ding K Z, Wang J L, Li J D, et al. Graph prototypical networks for few-shot learning on attributed networks. In: Proceedings of ACM International Conference on Information & Knowledge Management, 2020. 295–304
45 Lan L, Wang P H, Du X F, et al. Node classification on graphs with few-shot novel labels via meta transformed network embedding. In: Proceedings of Advances in Neural Information Processing Systems, 2020. 16520–16531
46 Wang N, Luo M N, Ding K Z, et al. Graph few-shot learning with attribute matching. In: Proceedings of International Conference on Information & Knowledge Management, 2020. 1545–1554
47 Huang K X, Zitnik M. Graph meta learning via local subgraphs. In: Proceedings of Advances in Neural Information Processing Systems, 2020. 5862–5874
48 Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. In: Proceedings of International Conference on Learning Representations, 2017. 1–12
49 Huang Y T, Chu L Y, Zhou Z R, et al. Personalized cross-silo federated learning on non-IID data. In: Proceedings of AAAI Conference on Artificial Intelligence, 2021. 7865–7873
50 Wei K, Li J, Ding M, et al. Federated learning with differential privacy: algorithms and performance analysis. IEEE Trans Inform Forensic Secur, 2020, 15: 3454–3469
51 Li J, Wei K, Ma C, et al. DP-GenFL: a local differentially private federated learning system through generative data. Sci China Inf Sci, 2023, 66: 189303
52 Bonawitz K, Ivanov V, Kreuter B, et al. Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2017. 1175–1191
53 Zhang C L, Li S Y, Xia J Z, et al. {BatchCrypt}: efficient homomorphic encryption for {Cross-Silo} federated learning. In: Proceedings of USENIX Annual Technical Conference (USENIX ATC 20), 2020. 493–506
54 Kang X, Yu G, Kong L, et al. FedTA: federated worthy task assignment for crowd workers. IEEE Trans Dependable Secure Comput, 2024, 21: 4098–4109
55 Tang J, Zhang J, Yao L M, et al. Arnetminer: extraction and mining of academic social networks. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008. 990–998
56 Bojchevski A, Günnemann S. Deep Gaussian embedding of graphs: unsupervised inductive learning via ranking. In: Proceedings of International Conference on Learning Representations, 2018. 1–12
57 McAuley J, Pandey R, Leskovec J. Inferring networks of substitutable and complementary products. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015. 785–794
58 Karypis G. Metis: Unstructured Graph Partitioning and Sparse Matrix Ordering System. Technical Report, 1997
59 Velickovic P, Cucurull G, Casanova A, et al. Graph attention networks. In: Proceedings of International Conference on Learning Representations, 2018. 1–12
60 van der Maaten L, Hinton G. Visualizing data using t-SNE. J Mach Learn Res, 2008, 9: 2579–2605