# Federated Graph Learning with Structure Proxy Alignment

Xingbo Fu
University of Virginia
Charlottesville, Virginia, USA
xf3av@virginia.edu

Zihan Chen
University of Virginia
Charlottesville, Virginia, USA
brf3rx@virginia.edu

Binchi Zhang
University of Virginia
Charlottesville, Virginia, USA
epb6gw@virginia.edu

Chen Chen
University of Virginia
Charlottesville, Virginia, USA
zrh6du@virginia.edu

Jundong Li
University of Virginia
Charlottesville, Virginia, USA
jundong@virginia.edu

## ABSTRACT

Federated Graph Learning (FGL) aims to learn graph learning models over graph data distributed in multiple data owners, which has been applied in various applications such as social recommendation and financial fraud detection. Inherited from generic Federated Learning (FL), FGL similarly has the data heterogeneity issue where the label distribution may vary significantly for distributed graph data across clients. For instance, a client can have the majority of nodes from a class, while another client may have only a few nodes from the same class. This issue results in divergent local objectives and impairs FGL convergence for node-level tasks, especially for node classification. Moreover, FGL also encounters a unique challenge for the node classification task: the nodes from a minority class in a client are more likely to have biased neighboring information, which prevents FGL from learning expressive node embeddings with Graph Neural Networks (GNNs). To grapple with the challenge, we propose FedSpray, a novel FGL framework that learns local class-wise structure proxies in the latent space and aligns them to obtain global structure proxies in the server. Our goal is to obtain the aligned structure proxies that can serve as reliable, unbiased neighboring information for node classification. To achieve this, FedSpray trains a global feature-structure encoder and generates unbiased soft targets with structure proxies to regularize local training of GNN models in a personalized way. We conduct extensive experiments over four datasets, and experiment results validate the superiority of FedSpray compared with other baselines. Our code is available at https://github.com/xbfu/FedSpray.

## CCS CONCEPTS

• **Computing methodologies → Distributed artificial intelligence**; **Neural networks**.

## KEYWORDS

Federated Learning, Graph Neural Network, Knowledge Distillation

**ACM Reference Format:**
Xingbo Fu, Zihan Chen, Binchi Zhang, Chen Chen, and Jundong Li. 2024. Federated Graph Learning with Structure Proxy Alignment. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671717

*KDD '24, August 25–29, 2024, Barcelona, Spain*
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671717

## 1 INTRODUCTION

Graph Neural Networks (GNNs) [46] are a prominent approach for learning expressive representations from graph-structured data. Typically, GNNs follow a message-passing mechanism, where the embedding of each node is computed by aggregating attribute information from its neighbors [11, 17, 44]. Thanks to their powerful capacity for jointly embedding attribute and graph structure information, GNNs have been widely adopted in a wide variety of applications, such as node classification [9, 12] and link prediction [2, 5]. The existing GNNs are mostly trained in a centralized manner where graph data is collected on a single machine before training. In the real world, however, a large number of graph data is generated by multiple data owners. These graph data cannot be assembled for training due to privacy concerns and commercial competitions [41], which prevents the traditional centralized manner from training powerful GNNs. Taking a financial system with four banks in Figure 1 as an example, each bank in the system has its local customer dataset and transactions between customers. As we take the customers in a bank as nodes and transactions between them as edges, the bank's local data can naturally form a graph. These banks aim to jointly train a GNN model for classification tasks, such as predicting a customer's occupation (i.e., *Doctor* or *Teacher*) without sharing their local data with each other.

Federated Learning (FL) [25] is a prevalent distributed learning scheme that enables multiple data owners (i.e., clients) to collaboratively train machine learning models under the coordination of a central server without sharing their private data. One critical challenge in FL is data heterogeneity, where data samples are not independent and identically distributed (i.e., non-IID) across the clients. For instance, assume that Bank A in Figure 1 locates in a community adjacent to a hospital. Then most customers in Bank A are therefore likely to be labeled as *Doctor* while only a few customers are from other occupations (e.g., *Teacher*). In contrast, Bank C adjoining a school has customers labeled mostly as *Teacher* and only a few as *Doctor*. Typically, the nodes from a class that claims the very large proportion of the overall data in a client are the *majority nodes* (e.g., *Doctor* in Bank A) while *minority nodes* (e.g., *Teacher* in Bank A) account for much fewer samples. The data heterogeneity issue results in divergent local objectives on the clients and consequently impairs the performance of FL [15]. A number

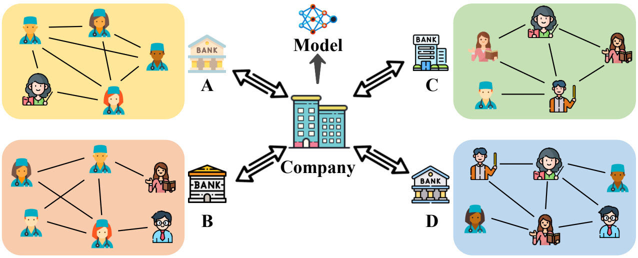


**Figure 1: An example of a financial system including four banks. The four banks aim to jointly train a model for predicting a customer's occupation (i.e., *Doctor* or *Teacher*) orchestrated by a third-party company over their local data while keeping their private data locally.**

of approaches have been proposed to address this issue, to name a few [20, 42, 45].

When we train GNNs over distributed graph data in a federated manner, however, the data heterogeneity issue can get much more severe. This results from a unique challenge in Federated Graph Learning (FGL) [10]: **the high heterophily of minority nodes**, i.e., their neighbors are mostly from other classes [35]. A majority node in a client (e.g., *Teacher* in Bank D) can benefit from the message-passing mechanism and obtain an expressive embedding as its neighbors are probably from the same class. On the contrary, a minority node in another client (e.g., *Teacher* in Bank A) may obtain biased information from its neighbors when they are from other classes (e.g., *Doctor* in Bank A). In FGL, this challenge is usually entangled with the data heterogeneity issue. As a result, the minority nodes will finally get underrepresented embeddings given adverse neighboring information and be more likely to be predicted as the major class, which results in unsatisfactory performance. Although a few studies have investigated the data heterogeneity issue about graph structures in FGL [38, 47], they did not fathom the divergent impact of neighboring information across clients for node classification.

To tackle the aforementioned challenges in FGL, we propose FedSpray, a novel FGL framework with structure proxy alignment in this study. The goal of FedSpray is to learn personalized GNN models for each client while avoiding underrepresented embeddings of the minority nodes in each client caused by their adverse neighboring information in FGL. To achieve this goal, we first introduce global class-wise structure proxies [7] which aim to provide nodes with informative, unbiased neighboring information, especially for those from the minority classes in each client. Moreover, FedSpray learns a global feature-structure encoder to obtain reliable soft targets that only depend on node features and aligned structure proxies. Then, FedSpray uses the soft targets to regularize local training of personalized GNN models via knowledge distillation [13]. We conduct extensive experiments over five graph datasets, and experimental results corroborate the effectiveness of the proposed FedSpray compared with other baselines.

We summarize the main contributions of this study as follows.

- **Problem Formulation.** We formulate and make an initial investigation on a unique issue of unfavorable neighboring information for minority nodes in FGL.
- **Algorithmic Design.** We propose a novel framework FedSpray to tackle the above problem in FGL. FedSpray aims to learn unbiased soft targets by a global feature-structure encoder with aligned class-wise structure proxies which provide informative, unbiased neighboring information for nodes and guide local training of personalized GNN models.
- **Experimental Evaluation.** We conduct extensive experiments over four graph datasets to verify the effectiveness of the proposed FedSpray. The experimental results demonstrate that our FedSpray consistently outperforms the state-of-the-art baselines.

## 2 PROBLEM FORMULATION

### 2.1 Preliminaries

*2.1.1* ***Notations.*** We use bold uppercase letters (e.g., $\mathbf{X}$) to represent matrices. For any matrix, e.g., $\mathbf{X}$, we denote its $i$-th row vector as $\mathbf{x}_i$. We use letters in calligraphy font (e.g., $\mathcal{V}$) to denote sets. $|\mathcal{V}|$ denotes the cardinality of set $\mathcal{V}$.

*2.1.2* ***Graph Neural Networks.*** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ denote an undirected attributed graph, where $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ is the set of $|\mathcal{V}|$ nodes, $\mathcal{E}$ is the edge set, and $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_x}$ is the node feature matrix. $d_x$ is the number of node features. Given each node $v_i \in \mathcal{V}$, $\mathcal{N}(v_i)$ denotes the set of its neighbors. The ground-truth label of each node $v_i \in \mathcal{V}$ can be denoted as a $d_c$-dimensional one-hot vector $\mathbf{y}_i$ where $d_c$ is the number of classes. The node homophily [23, 49] is defined as

$$h_i = \frac{|\{v_j | v_j \in \mathcal{N}(v_i) \text{ and } \mathbf{y}_j = \mathbf{y}_i\}|}{|\mathcal{N}(v_i)|}, \tag{1}$$

where $|\mathcal{N}(v_i)|$ denotes the degree of node $v_i$. Typically, an $L$-layer GNN model $f$ parameterized by $\theta$ maps each node to the outcome space via a message-passing mechanism [11, 17]. Specifically, each node $v_i$ aggregates information from its neighbors in the $l$-th layer of a GNN model by

$$\mathbf{h}_i^l = f_l(\mathbf{h}_i^{l-1}, \{\mathbf{h}_j^{l-1} : v_j \in \mathcal{N}(v_i)\}; \theta_l), \tag{2}$$

where $\mathbf{h}_i^l$ is the embedding of node $v_i$ after the $l$-th layer $f_l$, and $\theta_l$ is the parameters of the message-passing function in $f_l$. The raw feature of each node $v_i$ is used as the input layer, i.e., $\mathbf{h}_i^0 = \mathbf{x}_i$. For the node classification task, the node embedding $\mathbf{h}_i^L$ after the final layer is used to compute the predicted label distribution $\hat{\mathbf{y}}_i = \text{Softmax}(\mathbf{h}_i^L) \in \mathbb{R}^{d_p}$ by the softmax operator.

*2.1.3* ***Personalized FL.*** Given a set of $K$ clients, each client $k$ has its private dataset $\mathcal{D}^{(k)} = \{(\mathbf{x}_i^{(k)}, \mathbf{y}_i^{(k)})\}_{i=1}^{N^{(k)}}$, where $N^{(k)}$ is the number of samples in client $k$. The overall objective of the clients is

$$\min_{(\theta^{(1)},\theta^{(2)},\cdots,\theta^{(K)})} \sum_{k=1}^{K} \frac{N^{(k)}}{N} \mathcal{L}^{(k)}(\mathcal{D}^{(k)}; \theta^{(k)}), \tag{3}$$

where $\mathcal{L}^{(k)}(\theta^{(k)})$ is the local average loss (e.g., the cross-entropy loss) over local data in client $k$, and $N = \sum_{k=1}^{K} N^{(k)}$. Standard FL methods aim to learn a global model $\theta = \theta^{(1)} = \theta^{(2)} = \cdots = \theta^{(K)}$. As a representative method in FL, FedAvg [25] performs local updates in each client and uploads local model parameters to a

**Table 1: The statistics of the majority class and other minority classes in 7 clients from the PubMed dataset. *Majority* and *Minority* represent the majority class and other minority classes, respectively.**

| Client | Majority Class | Num. of Nodes | | Avg. Homophily | |
|---|---|---|---|---|---|
| | | Majority | Minority | Majority | Minority |
| 1 | 1 | **1,384** | 384 | **0.91** | 0.33 |
| 2 | 1 | **1,263** | 152 | **0.97** | 0.24 |
| 3 | 2 | **2,001** | 286 | **0.92** | 0.17 |
| 4 | 2 | **1,236** | 97 | **0.98** | 0.48 |
| 5 | 1 | **1,160** | 140 | **0.95** | 0.41 |
| 6 | 0 | **934** | 467 | **0.84** | 0.47 |
| 7 | 2 | **948** | 806 | **0.83** | 0.70 |

central server, where they are averaged by

$$\theta = \sum_{k=1}^{K} \frac{N^{(k)}}{N} \theta^{(k)} \tag{4}$$

during each round. However, a single global model may have poor performance due to the data heterogeneity issue in FL [19]. To remedy this, personalized FL [37] allows a customized $\theta^{(k)}$ in each client $k$ with better performance on local data while still benefiting from collaborative training.

## 2.2 Problem Setup

Given a set of $K$ clients, each client $k$ owns a local graph $\mathcal{G}^{(k)} = (\mathcal{V}^{(k)}, \mathcal{E}^{(k)}, \mathbf{X}^{(k)})$. For the labeled node set $\mathcal{V}_L^{(k)} \subset \mathcal{V}^{(k)}$ in client $k$, each node $v_i^{(k)} \in \mathcal{V}_L^{(k)}$ is associated with its label $\mathbf{y}_i^{(k)}$. The goal of these clients is to train personalized GNN models $f(\theta^{(k)})$ in each client $k$ for the node classification task while keeping their private graph data locally. Based on the aforementioned challenge and preliminary analysis, this study aims to enhance collaborative training by mitigating the impact of adverse neighboring information on node classification, especially for minority nodes.

# 3 MOTIVATION

In this section, we first conduct an empirical study on the PubMed dataset [31] to investigate the impact of divergent neighboring information across clients on minority nodes when jointly training GNNs in FGL. The observation from this study is consistent with our example in Figure 1 and motivates us to learn global structure proxies as favorable neighboring information. We then develop theoretical analysis to explain how aligning neighboring information across clients can benefit node classification tasks in FGL.

## 3.1 Empirical Observations

To better understand the divergent neighboring information across clients with its impact on the node classification task in FGL, we conduct preliminary experiments to compare the performance of federated node classification with MLP and GNNs as local models on the PubMed dataset [31]. Following the data partition strategy in previous studies [14, 51], we synthesize the distributed graph data by splitting each dataset into multiple communities via the Louvain algorithm [1]. We retain seven communities with the largest number of nodes; each community is regarded as an entire graph in a client.

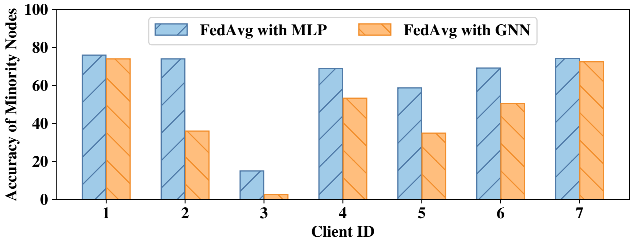


**Figure 2: Classification accuracy (%) of minority nodes in each client by training MLP and GNN via FedAvg over the PubMed dataset. Average accuracy for all nodes: 82.35% for MLP VS 87.06% for GNN.**

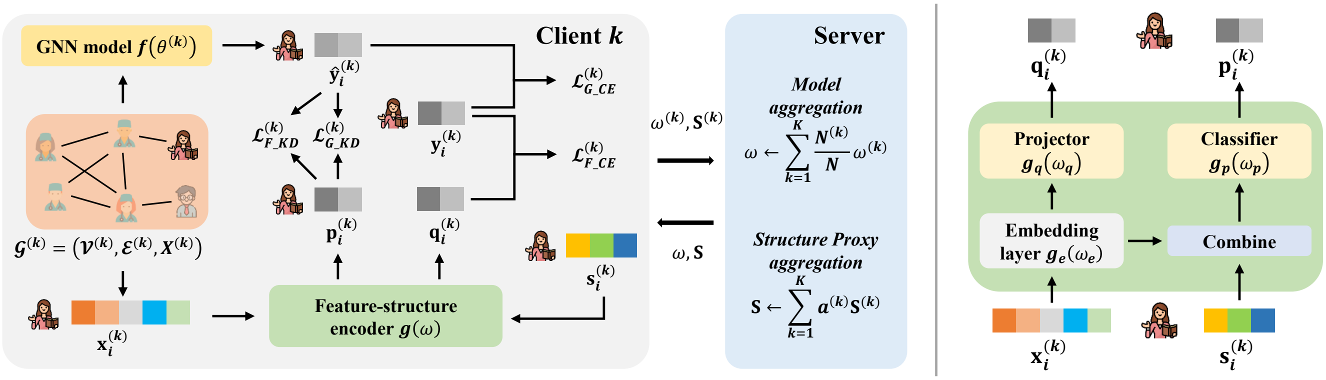
Table 1 shows the statistics of each client. According to Table 1, although one client may have the majority class different from another, the average node-level homophily of the majority class is consistently higher than that of the other classes for all the clients. For instance, the nodes in client 2 that do not belong to class 1 have only 24% neighbors from the same class on average. It means that the minority nodes will absorb unfavorable neighboring information via GNNs and probably be classified incorrectly.

To validate our conjecture, we perform collaborative training for MLPs and GNNs following the standard FedAvg [25] over the PubMed dataset. Figure 2 illustrates the classification accuracy of minority nodes in each client by MLPs and GNNs. We can observe that MLPs consistently perform better than GNNs on minority nodes across the clients, although GNNs have higher overall accuracy for all nodes. Given that MLPs and GNNs are trained over the same node label distribution, we argue that the performance gap on minority nodes results from aggregating adverse neighboring information from other classes via the message-passing mechanism in GNNs, especially from the majority class. On the contrary, MLPs only need node features and do not require neighboring information throughout the training; therefore, they can avoid predicting more nodes as the majority class.

## 3.2 Theoretical Motivation

According to the above empirical observations, minority nodes with the original neighboring information are more likely to be misclassified. One straightforward approach to this issue is enabling nodes to leverage favorable neighboring information from other clients for generating node embeddings. Specifically, we consider constructing global neighboring information in the feature space. The server collects neighboring feature vectors from each client and computes the global class-wise neighboring information via FedAvg [25]. We aim to theoretically investigate whether the global neighboring information can benefit node classification tasks when replacing the original neighbors of nodes. Following prevalent ways of graph modeling [8, 24, 40], we first generate random graphs in each client using a variant of contextual stochastic block model [40] with two classes.

*3.2.1* ***Random Graph Generation.*** The generative model generates a random graph in each client via the following strategy.


GNN model $f(\theta^{(k)})$
Client $k$
$\hat{\mathbf{y}}_i^{(k)}$
$\mathcal{L}_{G\_CE}^{(k)}$
$\mathcal{L}_{F\_KD}^{(k)}$
$\mathcal{L}_{G\_KD}^{(k)}$
$\mathbf{y}_i^{(k)}$
$\mathcal{L}_{F\_CE}^{(k)}$
$\mathbf{p}_i^{(k)}$
$\mathbf{q}_i^{(k)}$
$\mathbf{s}_i^{(k)}$
$\mathcal{G}^{(k)} = (\mathcal{V}^{(k)}, \mathcal{E}^{(k)}, \mathcal{X}^{(k)})$
$\mathbf{x}_i^{(k)}$
Feature-structure encoder $g(\omega)$
$\omega^{(k)}, \mathbf{S}^{(k)}$
$\omega, \mathbf{S}$
Server
Model aggregation
$\omega \leftarrow \sum_{k=1}^{K} \frac{N^{(k)}}{N} \omega^{(k)}$
Structure Proxy aggregation
$\mathbf{S} \leftarrow \sum_{k=1}^{K} a^{(k)} \mathbf{S}^{(k)}$
$\mathbf{q}_i^{(k)}$
$\mathbf{p}_i^{(k)}$
Projector $g_q(\omega_q)$
Classifier $g_p(\omega_p)$
Embedding layer $g_e(\omega_e)$
Combine
$\mathbf{x}_i^{(k)}$
$\mathbf{s}_i^{(k)}$

(a) The overview of FedSpray

(b) Feature-structure encoder

**Figure 3: (a) An overview of the proposed FedSpray. The backbone of FedSpray is personalized GNN models $f(\theta^{(k)})$. A global feature-structure encoder $g(\omega)$ with structure proxies $\mathbf{S}$ is also employed in FedSpray to tackle underrepresented node embeddings caused by adverse neighboring information in FGL. (b) An illustration of the feature-structure encoder in FedSpray.**

In the generated graph $\mathcal{G}^{(k)}$ in client $k$, the nodes are labeled by two classes $c_1$ and $c_2$. For each node $v_i^{(k)}$, its initial feature vector $\mathbf{x}_i^{(k)} \in \mathbb{R}^{d_x}$ is sampled from a Gaussian distribution $N(\boldsymbol{\mu}_1, \mathbf{I})$ if labeled as class $c_1$ or $N(\boldsymbol{\mu}_2, \mathbf{I})$ if labeled as class $c_2$ ($\boldsymbol{\mu}_1 \in \mathbb{R}^{d_x}$, $\boldsymbol{\mu}_2 \in \mathbb{R}^{d_x}$, and $\boldsymbol{\mu}_1 \neq \boldsymbol{\mu}_2$). For each client $k$, a neighbor of each node is from the majority with probability $p^{(k)}$ and from the minority with probability $1 - p^{(k)}$. The ratio of minority nodes and majority nodes is $q^{(k)}$. In our setting, we assume $\frac{1}{2} < p^{(k)} < 1$ and $0 < q^{(k)} < 1$. We denote each graph generated from the above strategy in client $k$ as $\mathcal{G}^{(k)} \sim \text{Gen}(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, p^{(k)}, q^{(k)})$.

*3.2.2* ***Better Separability with Global Neighboring Information.*** To figure out the influence of global neighboring information, we focus on the separability of the linear GNN classifiers with the largest margin when leveraging global neighboring information. Concretely, we aim to find the expected Euclidean distance from each class to the decision boundary of the optimal linear GNN classifier when it uses either the original neighboring information or the global neighboring information. We use *dist* and *dist′* to denote the expected Euclidean distances in these two scenarios, respectively. We summarize the results in the following proposition.

PROPOSITION 3.1. *Given a set of K clients, each client k owns a local graph* $\mathcal{G}^{(k)} \sim Gen(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, p^{(k)}, q^{(k)})$, *dist* $= \frac{||\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2||_2}{2}$, *which is smaller than dist′* $= \left(1 + \sum_{k=1}^{K}(1 - q^{(k)})(p^{(k)} - \frac{1}{2})\right) \frac{||\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2||_2}{2}$.

A detailed proof can be found in Appendix A. According to Proposition 3.1, we will have a larger expected distance *dist′* when using the global neighboring information. Typically, the larger the distance is, the smaller the misclassification probability is [24]. Therefore, the optimal linear GNN classifier will obtain better classification performance.

However, directly uploading neighboring feature vectors is implausible in FGL since it contains many sensitive raw features in the clients. To overcome this issue, we propose a novel framework FedSpray to learn global structure proxies in the latent space and elaborate on the details of FedSpray in Section 4.

## 4 METHODOLOGY

In this section, we present the proposed FedSpray in detail. Figure 3(a) illustrates an overview of FedSpray. The goal of FedSpray is to let the clients learn personalized GNN models over their private graph data while achieving higher performance by mitigating the impact of adverse neighboring information in GNN models. To reach this goal, FedSpray employs a lightweight global feature-structure encoder which learns class-wise structure proxies and aligns them on the central server. The feature-structure encoder generates reliable unbiased soft targets for nodes given their raw features and the aligned structure proxies to regularize local training of GNN models.

### 4.1 Personalized GNN Model

We first introduce personalized GNN models in FedSpray.

*4.1.1* ***GNN backbone Model.*** Considering their exceptional ability to model graph data, we use GNNs as the backbone of the proposed framework. In this study, we propose to learn GNN models for each client in a personalized manner to tackle the data heterogeneity issue in FGL. Specifically, the personalized GNN model $f(\theta^{(k)})$ in client $k$ outputs the predicted label distribution $\hat{\mathbf{y}}_i^{(k)}$ for each node $v_i^{(k)} \in \mathcal{V}_L^{(k)}$. Note that FedSpray is flexible. Any GNNs that follow the message-passing mechanism as the structure of Eq. (2) can be used as the backbone, such as GCN [17] and SGC [44].

*4.1.2* ***Loss formulation.*** During local training, $\theta^{(k)}$ can be updated by minimizing the cross-entropy loss between $\mathbf{y}_i^{(k)}$ and $\hat{\mathbf{y}}_i^{(k)}$ for each labeled node $v_i^{(k)} \in \mathcal{V}_L^{(k)}$

$$\mathcal{L}_{G\_CE}^{(k)} = \frac{1}{|\mathcal{V}_L^{(k)}|} \sum_{v_i^{(k)} \in \mathcal{V}_L^{(k)}} \text{CE}(\mathbf{y}_i^{(k)}, \hat{\mathbf{y}}_i^{(k)}), \tag{5}$$

where $\mathrm{CE}(\cdot,\cdot)$ denotes the cross-entropy loss. However, simply minimizing $\mathcal{L}_{G\_CE}^{(k)}$ can lead $\theta^{(k)}$ to overfitting during local training [19, 39]. In addition, the minority nodes are particularly prone to obtaining underrepresented embeddings due to biased neighboring information, as discussed above. To tackle this challenge, we propose to design an extra knowledge distillation term and use it to regularize local training of $\theta^{(k)}$. More concretely, we first employ the soft target $\mathbf{p}_i^{(k)} \in \mathbb{R}^{d_p}$ for each node $v_i^{(k)} \in \mathcal{V}^{(k)}$ generated by the global feature-structure encoder to guide local training of $\theta^{(k)}$ in client $k$. Typically, we hope $\mathbf{p}_i^{(k)}$ to be generated with unbiased neighboring information for node $v_i^{(k)}$ (we will elaborate on how to obtain proper $\mathbf{p}_i^{(k)}$ in Section 4.2). Then, we encourage $\hat{\mathbf{y}}_i^{(k)}$ to approximate $\mathbf{p}_i^{(k)}$ by minimizing the discrepancy between $\mathbf{p}_i^{(k)}$ and $\hat{\mathbf{y}}_i^{(k)}$ for each node $v_i^{(k)} \in \mathcal{V}^{(k)}$ in client $k$. Specifically, we achieve this via knowledge distillation [13] as

$$\mathcal{L}_{G\_KD}^{(k)} = \frac{1}{|\mathcal{V}^{(k)}|} \sum_{v_i^{(k)} \in \mathcal{V}^{(k)}} \mathrm{KL}(\mathbf{p}_i^{(k)} \| \hat{\mathbf{y}}_i^{(k)}), \tag{6}$$

where $\mathrm{KL}(\cdot\|\cdot)$ is to compute the Kullback-Leibler divergence (KL-divergence). Therefore, the overall loss for training $\theta^{(k)}$ in client $k$ can be formulated by combining the two formulations together

$$\mathcal{L}_{G}^{(k)} = \mathcal{L}_{G\_CE}^{(k)} + \lambda_1 \mathcal{L}_{G\_KD}^{(k)}, \tag{7}$$

where $\lambda_1$ is a predefined hyperparameter that controls the contribution of the knowledge distillation term in $\mathcal{L}_G^{(k)}$. When $\lambda_1$ is set as 0, FedSpray will be equivalent to training GNN models individually in each client.

## 4.2 Global Feature-Structure Encoder with Structure Proxies

In this subsection, we will elucidate our design for the global feature-structure encoder and class-wise structure proxies in FedSpray. The feature-structure encoder aims to generate a reliable soft target (i.e., $\mathbf{p}_i^{(k)}$) for each node with its raw features and structure proxy.

### 4.2.1 *Structure Proxies.*

As discussed above, a minority node can obtain adverse neighboring information from its neighbors via the message-passing mechanism, given its neighbors are probably from other classes. To mitigate this issue, we propose to learn unbiased class-wise structure proxies in FedSpray, providing favorable neighboring information for each node. Here, we formulate each structure proxy in a vectorial form. Let $\mathbf{S} \in \mathbb{R}^{d_c \times d_s}$ denote class-wise structure proxies, and each row $\mathbf{s}_j \in \mathbf{S}$ denotes the $d_s$-dimensional structure proxy of the $j$-th node class. For each node $v_i^{(k)} \in \mathcal{V}_L^{(k)}$, its structure proxy $\mathbf{s}_i^{(k)}$ will be $\mathbf{s}_j$ if it is from the $j$-th class. Then, the structure proxies will be used as the input of the feature-structure encoder.

### 4.2.2 *Feature-Structure Encoder.*

In FedSpray, we employ a lightweight feature-structure encoder to generate a reliable soft target for a node with its raw feature and structure proxy as the input. Figure 3(b) illustrates our design for the feature-structure encoder. Let $g(\omega)$ denote the feature-structure encoder $g$ parameterized by $\omega$. Given a node $v_i^{(k)} \in \mathcal{V}_L^{(k)}$, the feature-structure encoder $g$ generates its soft target $\mathbf{p}_i^{(k)}$ with its feature vector $\mathbf{x}_i^{(k)}$ and structure proxy $\mathbf{s}_i^{(k)}$ by

$$\mathbf{p}_i^{(k)} = g(\mathbf{x}_i^{(k)}, \mathbf{s}_i^{(k)}; \omega). \tag{8}$$

**Fusion of node features and structure proxies.** Here, the problem is to determine a proper scheme for fusing a node's raw feature and its structure proxy in the feature-structure encoder. A straightforward way is to combine $\mathbf{x}_i^{(k)}$ and $\mathbf{s}_i^{(k)}$ together as the input of the feature-structure encoder. Ideally, $\mathbf{s}_i^{(k)}$ can serve as surrogate neighboring information of node $v_i^{(k)}$ in the feature space. In this case, it requires $\mathbf{s}_i^{(k)}$ to have the same dimension as that of $\mathbf{x}_i^{(k)}$. However, this brings us a new challenge: when $\mathbf{x}_i^{(k)}$ is of high dimension in graph data (e.g., 500 for PubMed [31]), directly learning high-dimensional $\mathbf{s}_i^{(k)}$ in the feature space will be intractable. Considering this, we propose to learn $\mathbf{s}_i^{(k)}$ in the latent space instead. Specifically, we split the feature-structure encoder into an embedding layer $g_e(\omega_e)$ and a classifier $g_p(\omega_p)$. The embedding layer first maps the raw feature $\mathbf{x}_i^{(k)}$ of a node $\mathbf{v}_i^{(k)} \in \mathcal{V}^{(k)}$ into the latent space to obtain its low-dimensional feature embedding $\mathbf{e}_i^{(k)}$. Then we combine the feature embedding $\mathbf{e}_i^{(k)}$ and the structure proxy $\mathbf{s}_i^{(k)}$ together as the input of the classifier to get the soft target $\mathbf{p}_i^{(k)}$. Mathematically, we can formulate this procedure as

$$\mathbf{p}_i^{(k)} = g(\mathbf{x}_i^{(k)}, \mathbf{s}_i^{(k)}; \omega) = g_p(\mathrm{Combine}(\mathbf{e}_i^{(k)}, \mathbf{s}_i^{(k)}); \omega_p) \tag{9}$$

where $\mathbf{e}_i^{(k)} = g_e(\mathbf{x}_i^{(k)}; \omega_e)$. Here, $\mathrm{Combine}(\cdot,\cdot)$ is the operation to combine $\mathbf{e}_i^{(k)}$ and $\mathbf{s}_i^{(k)}$ together (e.g., addition).

**Structure proxies for unlabeled nodes.** The feature-structure encoder can generate soft targets only for labeled nodes by Eq. (9) because the structure proxy $\mathbf{s}_i^{(k)}$ requires the ground-truth label information of node $\mathbf{v}_i^{(k)}$. To better regularize local training of the GNN model, we need to obtain soft targets for unlabeled nodes and use them to compute $\mathcal{L}_{G\_KD}^{(k)}$ by Eq. (6). To achieve this, we design a projector $g_q(\omega_q)$ in the feature-structure encoder. It has the same structure as the classifier $g_p$. The difference is that the projector $g_q$ generates soft targets only based on feature embeddings. Specifically, we can obtain the soft label $\mathbf{q}_i^{(k)}$ for each node $\mathbf{v}_i^{(k)} \in \mathcal{V}^{(k)} \setminus \mathcal{V}_L^{(k)}$ with its feature embedding $\mathbf{e}_i^{(k)}$ by

$$\mathbf{q}_i^{(k)} = g_q(\mathbf{e}_i^{(k)}; \omega_q). \tag{10}$$

Therefore, we obtain the structure proxy $\mathbf{s}_i^{(k)} = \langle \mathbf{q}_i^{(k)}, \mathbf{S} \rangle$ by computing the product of $\mathbf{q}_i^{(k)}$ and $\mathbf{S}$ for each node $\mathbf{v}_i^{(k)} \in \mathcal{V}^{(k)} \setminus \mathcal{V}_L^{(k)}$. Since $\mathbf{q}_i^{(k)}$ is normalized by the softmax operation, the inner product can also be viewed as the weighted average of $\mathbf{S}$.

### 4.2.3 *Loss formulation.*

During local training, we aim to update $\omega = \{\omega_e, \omega_p, \omega_q\}$ and $\mathbf{S}$ using both ground-truth labels and predictions from the GNN model. Specifically, we formulate the overall loss for training $\omega$ and $\mathbf{S}$ in client $k$ as

$$\mathcal{L}_F^{(k)} = \mathcal{L}_{F\_CE}^{(k)} + \lambda_2 \mathcal{L}_{F\_KD}^{(k)}, \tag{11}$$

where $\lambda_2$ is a hyperparameter. Here $\mathcal{L}^{(k)}_{F\_CE}$ is the average cross-entropy loss between $\mathbf{y}^{(k)}_i$ and $\mathbf{q}^{(k)}_i$ for each node $v^{(k)}_i \in \mathcal{V}^{(k)}_L$

$$\mathcal{L}^{(k)}_{F\_CE} = \frac{1}{|\mathcal{V}^{(k)}_L|} \sum_{v^{(k)}_i \in \mathcal{V}^{(k)}_L} \mathrm{CE}(\mathbf{y}^{(k)}_i, \mathbf{q}^{(k)}_i). \tag{12}$$

$\mathcal{L}^{(k)}_{F\_KD}$ is the average KL-divergence between $\mathbf{p}^{(k)}_i$ and $\hat{\mathbf{y}}^{(k)}_i$ to encourage $\mathbf{p}^{(k)}_i$ to approach $\hat{\mathbf{y}}^{(k)}_i$ for each node $v^{(k)}_i \in \mathcal{V}^{(k)}_L$

$$\mathcal{L}^{(k)}_{F\_KD} = \frac{1}{|\mathcal{V}^{(k)}_L|} \sum_{v^{(k)}_i \in \mathcal{V}^{(k)}_L} \mathrm{KL}(\hat{\mathbf{y}}^{(k)}_i \| \mathbf{p}^{(k)}_i). \tag{13}$$

## 4.3 Server Update

As stated above, FedSpray will learn the feature-structure encoder and the structure proxies globally. In this subsection, we present the global update in the central server for the feature-structure encoder and the structure proxies, respectively.

*4.3.1* ***Update global feature-structure encoder.*** During each round $r$, the server performs weighted averaging of local feature-structure encoders following the standard FedAvg [25] with each coefficient determined by the local node size

$$\omega_r \leftarrow \sum_{k=1}^{K} \frac{N^{(k)}}{N} \omega^{(k)}_r. \tag{14}$$

*4.3.2* ***Structure proxy alignment.*** Instead of using the local node size, we propose to assign higher weights to majority classes than minority classes for structure proxy alignment. More specifically, the server updates global structure proxy $\mathbf{s}_{j,r} \in \mathbf{S}_r$ during round $r$ by

$$\mathbf{s}_{j,r} \leftarrow \sum_{k=1}^{K} \frac{a^{(k)}_j}{a_j} \mathbf{s}^{(k)}_{j,r}, \tag{15}$$

where $a^{(k)}_j$ is the ratio of nodes from the $j$-th class among $\mathcal{V}^{(k)}_L$ in client $k$ and $a_j = \sum_{k=1}^{K} a^{(k)}_j$.

## 4.4 Overall Algorithm

Algorithm 1 shows the overall algorithm of the proposed FedSpray. During each round, each client performs local updates with two phases. In Phase 1, each client trains its personalized GNN models for $E$ epochs. We first compute $\mathbf{p}^{(k)}_i$ for node $v^{(k)}_i$ by the global feature-structure encoder $g(\omega_{r-1})$ with its feature $\mathbf{x}^{(k)}_i$ and corresponding structure proxy $\mathbf{s}^{(k)}_i$ (line 5). Then $\mathbf{p}^{(k)}_i$ is utilized to compute $\mathcal{L}^{(k)}_G$ (line 9) for training the GNN model (line 10). In Phase 2, the feature-structure encoder and structure proxies will be optimized for $E$ epochs. In client $k$, we first obtain $\hat{\mathbf{y}}^{(k)}_i$ for node $v^{(k)}_i$ by the up-to-date GNN model (line 14). $\hat{\mathbf{y}}^{(k)}_i$ for node $v^{(k)}_i$ will be used to compute $\mathcal{L}^{(k)}_F$ (line 19). Then we update $\omega^{(k)}_{r,t}$ and $\mathbf{s}^{(k)}_i$ via gradient descent (line 20-21). At the end of each round, $\mathbf{s}_j \in \mathbf{S}^{(k)}_r$ will be updated by averaging $\mathbf{s}^{(k)}_i$ of nodes from the $j$-th class (line 23). At the end of each round, the local feature-structure encoder and structure proxies will be sent to the central server for training in the next round (line 24). In the central server, FedSpray updates the feature-structure encoder by Eq. (14) and aligns structure proxies by Eq. (15).

**Algorithm 1** FedSpray

```
Input: initial personalized $\theta^{(k)}$ for each client $k$, global $\omega_0$ and $\mathbf{S}_0$
  for each round $r = 1, \cdots, R$ do
    for each client $k$ in parallel do
      $\omega^{(k)}_r, \mathbf{S}^{(k)}_r \leftarrow$ LocalUpdate $(\omega_{r-1}, \mathbf{S}_{r-1})$
    end for
    Update $\omega_r$ by Eq. (14)
    Update $\mathbf{S}_r$ by Eq. (15)
  end for

LocalUpdate$(\omega_{r-1}, \mathbf{S}_{r-1})$:
1:  ================= Phase 1 ====================
2:  $\mathbf{e}^{(k)}_i = g_e(\mathbf{x}^{(k)}_i; \omega_{e,r-1})$
3:  $\mathbf{q}^{(k)}_i = g_q(\mathbf{e}^{(k)}_i; \omega_{q,r-1})$
4:  Compute local $\mathbf{s}^{(k)}_i$ from $\mathbf{S}_{r-1}$
5:  $\mathbf{p}^{(k)}_i = g_p(\mathrm{Combine}(\mathbf{e}^{(k)}_i, \mathbf{s}^{(k)}_i); \omega_{p,r-1})$
6:  $\theta^{(k)}_r = \theta_{r-1}$
7:  for $t = 1, \cdots, E$ do
8:      $\hat{\mathbf{y}}^{(k)}_i = f(\mathbf{x}^{(k)}_i, \mathcal{G}^{(k)}; \theta^{(k)}_r)$
9:      Compute $\mathcal{L}^{(k)}_G$ by Eq. (7) using $\mathbf{p}^{(k)}_i$
10:     Update the local GNN model $\theta^{(k)}_r \leftarrow \theta^{(k)}_r - \eta_f \nabla \mathcal{L}^{(k)}_G$
11: end for
12: ================= Phase 2 ====================
13: $\omega^{(k)}_r = \omega_{r-1}$
14: $\hat{\mathbf{y}}^{(k)}_i = f(\mathbf{x}^{(k)}_i, \mathcal{G}^{(k)}; \theta^{(k)}_r)$
15: for $t = 1, \cdots, E$ do
16:     $\mathbf{e}^{(k)}_i = g_e(\mathbf{x}^{(k)}_i; \omega^{(k)}_{e,r})$
17:     $\mathbf{q}^{(k)}_i = g_q(\mathbf{e}^{(k)}_i; \omega^{(k)}_{q,r})$
18:     $\mathbf{p}^{(k)}_i = g_p(\mathrm{Combine}(\mathbf{e}^{(k)}_i, \mathbf{s}^{(k)}_i); \omega^{(k)}_{p,r})$
19:     Compute $\mathcal{L}^{(k)}_F$ by Eq. (11) using $\hat{\mathbf{y}}^{(k)}_i$
20:     Update the local feature-structure encoder
        $\omega^{(k)}_r \leftarrow \omega^{(k)}_r - \eta_g \nabla \mathcal{L}^{(k)}_F$
21:     Update the local structure proxy
        $\mathbf{s}^{(k)}_i \leftarrow \mathbf{s}^{(k)}_i - \eta_s \nabla \mathcal{L}^{(k)}_F$
22: end for
23: Update $\mathbf{s}_j \in \mathbf{S}^{(k)}_r$ by averaging $\mathbf{s}^{(k)}_i$ of nodes from class $j$
24: return $\omega^{(k)}_r, \mathbf{S}^{(k)}_r$
```

## 4.5 Discussion

FedSpray exhibits superior advantages from various perspectives, including communication efficiency, privacy preservation, and computational cost. We provide an in-depth discussion about FedSpray's principal properties as follows.

*4.5.1* ***Privacy Preservation.*** The proposed FedSpray uploads the parameters of local feature-structure encoders following the prevalent frameworks in FL [18–20]. Here, we mainly discuss the privacy

**Table 2: Classification accuracy (Average±Std) of FedSpray and other baselines on node classification over four datasets. *Overall* and *Minority* represent all nodes and minority nodes in the test sets, respectively.**

| Dataset | Method | GCN | | SGC | | GraphSAGE | |
|---|---|---|---|---|---|---|---|
| | | Overall | Minority | Overall | Minority | Overall | Minority |
| PubMed | Local | 87.49 ± 0.24 | 51.00 ± 1.20 | 86.27 ± 0.34 | 40.83 ± 0.93 | 86.86 ± 0.26 | 48.66 ± 1.14 |
| | Fedavg | 87.06 ± 0.61 | 55.77 ± 0.90 | 82.23 ± 1.89 | 56.21 ± 1.38 | 85.92 ± 0.87 | 61.35 ± 2.76 |
| | APFL | 86.44 ± 0.66 | 49.25 ± 2.07 | 83.10 ± 0.34 | 28.25 ± 5.69 | 86.23 ± 0.55 | 45.41 ± 1.50 |
| | GCFL | 86.74 ± 0.69 | 48.85 ± 3.25 | 71.81 ± 8.22 | 51.10 ± 8.66 | 85.35 ± 0.46 | 45.60 ± 2.53 |
| | FedStar | 81.25 ± 0.67 | 12.01 ± 2.89 | 82.06 ± 1.32 | 19.24 ± 7.71 | 80.57 ± 1.21 | 7.78 ± 7.01 |
| | FedLit | 57.95 ± 3.82 | 47.39 ± 2.84 | 84.82 ± 0.72 | 57.54 ± 1.76 | 70.73 ± 7.61 | 57.15 ± 8.35 |
| | **FedSpray** | **87.71 ± 0.65** | **62.12 ± 2.73** | **87.13 ± 1.41** | **59.23 ± 1.25** | **87.02 ± 1.01** | **61.59 ± 0.96** |
| WikiCS | Local | 81.43 ± 0.58 | 41.36 ± 1.78 | 81.66 ± 0.62 | 40.02 ± 1.41 | 81.57 ± 0.35 | 42.93 ± 1.85 |
| | Fedavg | 80.53 ± 0.74 | 35.48 ± 2.52 | 79.90 ± 0.60 | 34.24 ± 1.17 | 80.23 ± 0.20 | 47.60 ± 0.99 |
| | APFL | 79.81 ± 0.72 | 38.33 ± 5.43 | 78.46 ± 0.63 | 27.40 ± 1.52 | 80.54 ± 0.64 | 42.16 ± 1.21 |
| | GCFL | 75.79 ± 1.56 | 36.94 ± 1.80 | 74.85 ± 1.65 | 29.08 ± 0.69 | 73.34 ± 3.24 | 37.79 ± 2.90 |
| | FedStar | 75.61 ± 0.53 | 16.72 ± 3.23 | 76.95 ± 0.73 | 21.90 ± 2.82 | 74.48 ± 0.51 | 10.66 ± 1.71 |
| | FedLit | 49.08 ± 4.98 | 29.85 ± 2.85 | 56.18 ± 9.39 | 32.79 ± 3.73 | 60.37 ± 4.33 | 36.90 ± 4.32 |
| | **FedSpray** | **81.51 ± 0.45** | **47.43 ± 1.31** | **81.87 ± 0.59** | **46.60 ± 0.00** | **81.93 ± 0.30** | **52.04 ± 0.51** |
| Physics | Local | 94.62 ± 0.16 | 72.75 ± 0.73 | 94.82 ± 0.28 | 76.24 ± 9.07 | 94.14 ± 0.30 | 69.50 ± 0.97 |
| | Fedavg | 94.13 ± 0.40 | 66.45 ± 2.28 | 94.40 ± 0.25 | 66.58 ± 1.01 | 94.60 ± 0.34 | 74.27 ± 0.95 |
| | APFL | 94.27 ± 0.20 | 72.83 ± 3.73 | 94.52 ± 0.27 | 69.27 ± 1.67 | 84.31 ± 3.76 | 38.65 ± 7.33 |
| | GCFL | 88.97 ± 2.61 | 60.90 ± 2.04 | 94.02 ± 0.29 | 66.54 ± 1.87 | 80.71 ± 3.91 | 50.22 ± 5.20 |
| | FedStar | 89.86 ± 0.43 | 33.44 ± 3.27 | 91.37 ± 0.40 | 45.27 ± 4.73 | 89.78 ± 0.41 | 32.91 ± 3.52 |
| | FedLit | 85.11 ± 2.58 | 60.57 ± 2.42 | 87.57 ± 1.47 | 61.96 ± 0.81 | 86.68 ± 0.27 | 66.36 ± 0.88 |
| | **FedSpray** | **95.59 ± 0.24** | **80.98 ± 1.39** | **95.08 ± 0.32** | **82.43 ± 1.62** | **94.73 ± 0.37** | **83.26 ± 1.25** |
| Flickr | Local | 43.18 ± 0.55 | 25.96 ± 1.94 | 46.82 ± 0.93 | 25.39 ± 1.46 | 49.72 ± 0.85 | 25.25 ± 1.70 |
| | Fedavg | 44.53 ± 1.36 | 26.45 ± 0.46 | 47.03 ± 1.39 | 27.24 ± 2.52 | 47.51 ± 1.40 | 26.13 ± 0.82 |
| | APFL | 32.27 ± 3.58 | 19.44 ± 6.16 | 46.93 ± 0.50 | 23.50 ± 0.49 | 34.59 ± 2.83 | 18.39 ± 2.96 |
| | GCFL | 47.31 ± 1.29 | 19.71 ± 2.20 | 46.56 ± 1.71 | 26.48 ± 3.60 | 44.84 ± 2.10 | 16.76 ± 2.45 |
| | FedStar | 47.73 ± 0.85 | 13.82 ± 3.05 | 48.45 ± 0.58 | 14.59 ± 3.39 | 46.36 ± 1.04 | 11.33 ± 4.38 |
| | FedLit | 45.38 ± 1.73 | 23.62 ± 8.74 | 49.62 ± 0.36 | 24.46 ± 0.81 | 44.06 ± 2.26 | 18.12 ± 2.30 |
| | **FedSpray** | **48.21 ± 1.03** | **29.72 ± 0.75** | **50.07 ± 0.75** | **28.46 ± 2.12** | **51.45 ± 0.72** | **27.52 ± 0.42** |

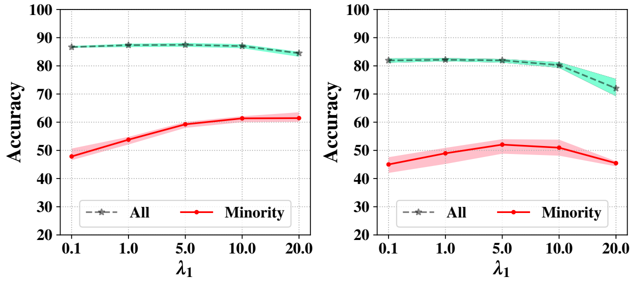
concern about uploading local structure proxies first. In fact, structure proxies naturally protect data privacy. First, they are synthetic 1D vectors to provide high-quality neighboring information in the latent space. In other words, they do not possess any raw feature information. Second, they are generated by averaging the structure proxies from the same class, which is an irreversible operation. Moreover, we can employ various privacy-preserving techniques to further improve confidentiality.

*4.5.2* ***Communication Efficiency.*** The proposed FedSpray requires clients to upload local feature-structure encoders and structure proxies. As we introduced above, the feature-structure encoder is a relatively lightweight model. As for structure proxies, their size is generally much smaller than that of model parameters given $d_s \ll d_x$. In addition, we can further reduce the number of uploaded parameters by setting smaller $d_s$.

*4.5.3* ***Computational Cost.*** The additional computational cost in FedSpray is mainly on local updates for the feature-structure encoder and structure proxies. Compared with GNN models, the feature-structure encoder and structure proxies require fewer operations for updating parameters. Training GNN models is usually time-consuming since GNN models need to aggregate node information via the message-passing mechanism during the forward pass [52]. However, the feature-structure encoder only incorporates node features and structure proxies with fully connected layers to obtain soft targets. Therefore, the time complexity of local updates for the feature-structure encoder and structure proxies will be smaller than GNN models. Let $N$, $d_x$, and $E$ denote the number of nodes of the local graph in a client, the number of node features, and the number of edges, respectively. Considering a 2-layer GCN model with hidden size $d_h$, its computational complexity is approximately $O(Nd_hd_x + Ed_x)$. Similarly, we can conclude that the computational complexity of the feature-structure encoder with the $d_s$-dimensional structure proxy is about $O(Nd_sd_x)$, apparently smaller than the GCN model when we set $d_h = d_s$. Therefore, the feature-structure encoder in FedSpray does not introduce significant extra computational costs compared with FedAvg using GCN. Furthermore, setting a smaller $d_s$ can also reduce computation costs.

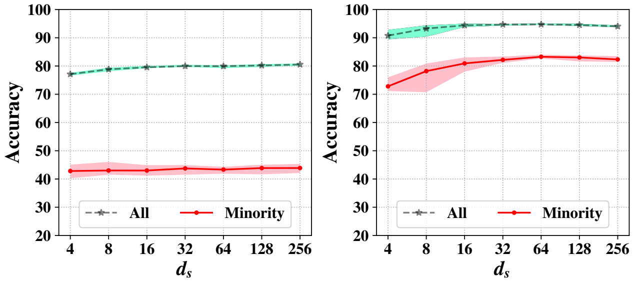
## 5 EXPERIMENTS

In this section, we conduct empirical experiments to demonstrate the effectiveness of the proposed framework FedSpray and perform detailed analysis of FedSpray.



(a) PubMed with GraphSAGE (b) WikiCS with GraphSAGE

**Figure 4: Classification accuracy (%) of FedSpray on all nodes and minority nodes in the test sets with different values of $\lambda_1$ over (a) PubMed and (b) WikiCS with GraphSAGE.**



(a) WikiCS with GCN (b) Physics with GCN

**Figure 5: Classification accuracy (%) of FedSpray on all nodes and minority nodes in the test sets with different $d_s$ over (a) WikiCS and (b) Physics with GCN.**

## 5.1 Experiment Setup

*5.1.1* ***Datasets.*** We synthesize the distributed graph data based on four common real-world datasets from various domains, i.e., PubMed [31], WikiCS [26], Coauthor Physics [33], and Flickr [50]. We follow the strategy in Section 3.1 to simulate the distributed graph data and summarize the statistics and basic information about the datasets in Appendix B.1. We randomly select nodes in clients and let 40% for training, 30% for validation, and the remaining for testing. We report the average classification accuracy for all nodes and minority nodes over the clients for five random repetitions.

*5.1.2* ***Baselines.*** We compare FedSpray with six baselines including (1) **Local** where each client train its GNN model individually; (2) **FedAvg** [25], the standard FL algorithm; (3) **APFL** [6], an adaptive approach in personalized FL; (4) **GCFL** [47], (5) **FedStar** [38], and (6) **FedLit** [48], three state-of-the-art FGL methods. More details about the above baselines can be found in Appendix B.2.

*5.1.3* ***Hyperparameter setting.*** As stated previously, FedSpray is compatible with most existing GNN architectures. In the experiments, we adopt three representative ones as backbone models: GCN [17], SGC [44], and GraphSAGE [11]. Each GNN model includes two layers with a hidden size of 64. The size of feature embeddings and structure proxies is also set as 64. Therefore, the feature-structure encoder has similar amounts of parameters with GNN models. Each component in the feature-structure encoder is implemented with one layer. We use an Adam optimizer [16] with learning rates of 0.003 for the global feature-structure encoder and personalized GNN models, 0.02 for structure proxies. The two hyperparameters $\lambda_1$ and $\lambda_2$ are set as 5 and 1, respectively. We run all the methods for 300 rounds, and the local epoch is set as 5.

## 5.2 Effectiveness of FedSpray

We first show the performance of FedSpray and other baselines on node classification over the four datasets with three backbone GNN models. Table 2 reports the average classification accuracy on all nodes and minority nodes in the test set across clients.

First, we analyze the results of overall accuracy on all test nodes. According to Table 2, our FedSpray consistently outperforms all the baselines on node classification accuracy for overall test nodes across clients. Local and FedAvg achieve comparable performance over the four datasets. In the meantime, APFL does not surpass Local and FedAvg. As for FGL methods, GCFL, FedStar, FedLit fail to show remarkable performance gain. Although GCFL and FedStar tackle the data heterogeneity issue of graph structures across clients in FGL, they do not take the node-level heterophily into account. While FedLit models latent link types between nodes via multi-channel GNNs, it involves more GNN parameters that are hard to be well trained within limited communication rounds.

Second, we analyze the results of accuracy on minority nodes in the test set. Note that FedSpray aims to learn reliable unbiased structure information for guiding local training of personalized GNN models, particularly for minority nodes. We can observe that FedSpray outperforms all the baselines by a notable margin. Even though Local and FedAvg achieve comparable performance on overall test nodes, they show different accuracy results on minority nodes. Among the three FGL methods, FedStar encounters significant performance degradation on minority nodes since the design of structure embeddings in FedStar does not provide beneficial neighboring information for node-level tasks.

## 5.3 Analysis of FedSpray

*5.3.1* ***Influence of hyperparameter*** $\lambda_1$. The hyperparameter $\lambda_1$ controls the contribution of the regularization term in $\mathcal{L}_G(k)$. We conduct the sensitivity analysis on $\lambda_1$ in FedSpray. Figure 4 reports the classification accuracy of FedSpray on all nodes and test nodes in the test sets with different values of $\lambda_1$ over PubMed (left) and WikiCS (right) with GraphSAGE. The accuracy on all nodes remains high when $\lambda_1$ is relatively small (i.e., $\lambda_1 = 0.1, 1, 5$). However, the accuracy of minority nodes will decrease when $\lambda_1$ is too small because the feature-structure encoder cannot sufficiently regularize local training of GNN models with too small $\lambda_1$. When $\lambda_1$ gets too large, the accuracy of all nodes decreases in both figures. In this case, the regularization term weighs overwhelmingly in the loss for training GNN models; then GNN models cannot be sufficiently trained with label information. According to the above observations, we will recommend 10 for PubMed with GraphSAGE and 5 for WikiCS with GraphSAGE as the best setting for $\lambda_1$.

*5.3.2* ***Influence to structure proxy dimension.*** Since FedSpray incorporates structure proxies in the feature-structure encoder, we may set a different dimension $d_s$ of structure proxies. We evaluate

**Table 3: Classification accuracy (Average±Std) of FedSpray with S = 0 over PubMed and Physics with GCN.**

| Dataset | Method | Overall | Minority |
|---|---|---|---|
| PubMed | FedSpray | **87.71 ± 0.65** | **62.12 ± 2.73** |
| | FedSpray (**S** = **0**) | 77.11 ± 0.43 | 42.01 ± 0.81 |
| Physics | FedSpray | **95.59 ± 0.24** | **80.98 ± 1.39** |
| | FedSpray (**S** = **0**) | 93.23 ± 0.27 | 72.57 ± 0.38 |

the performance of FedSpray with different values of $d_s$ while fixing the hidden dimension of the GNN model as 64. Figure 5 demonstrates the classification accuracy of FedSpray on all nodes and test nodes in the test sets with different values of $d_s$ over WikiCS (left) and Physics (right) with GCN as the backbone. We can observe that FedSpray can obtain comparable accuracy with $d_s$ smaller than 64 (e.g., $d_s$ = 32). In the meantime, FedSpray does not obtain significant performance gain when $d_s$ is larger than 64. From the above observation, we can reduce communication and computation costs by setting $d_s$ a smaller value such as 32.

*5.3.3* ***Effectiveness of structure proxies.*** In this study, we design structure proxies in FedSpray to serve as global unbiased neighboring information for guiding local training of GNN models. To validate the effectiveness of structure proxies, we investigate the performance of the proposed framework when structure proxies are removed. Specifically, we set class-wise structure proxies **S** as **0** consistently during training. We report the performance of FedSpray with **S** = **0** over PubMed and WikiCS in Table 3. According to Table 3, we can observe that FedSpray suffers from significant performance degradation when removing structure proxies. It suggests that structure proxies play a significant role in FedSpray. Without them, the feature-structure encoder generates soft targets only based on node features [52]. In this case, the soft labels can be unreliable when node labels are not merely dependent on node features and, therefore, provide inappropriate guidance on local training of personalized GNN models in FedSpray.

*5.3.4* ***More Experimental Results.*** Due to the page limit, we provide experimental results of FedSpray with varying local epochs in Appendix B.3.

# 6 RELATED WORK

## 6.1 Federated Learning

Recent years have witnessed the booming of techniques in FL and its various applications in a wide range of domains, such as computer vision [3, 28], healthcare [21, 36], and social recommendation [22, 43]. The most important challenge in FL is data heterogeneity across clients (i.e., the non-IID problem). A growing number of studies have been proposed to mitigate the impact of data heterogeneity. For instance, FedProx [20] adds a proximal term to the local training loss to keep the updated parameters close to the global model. Moon [18] uses a contrastive loss to increase the distance between the current and previous local models. FedDecorr [34] mitigates dimensional collapse to prevent representations from residing in a lower-dimensional space. In the meantime, a battery of studies proposed personalized model-based methods. For example, pFedHN [32] trains a central hypernetwork to output a unique personalized model for each client. APFL [6] learns a mixture of local and global models as the personalized model. FedProto [39] and FedProc [27] utilize the prototypes to regularize local model training. FedBABU [28] proposes to keep the global classifier unchanged during the feature representation learning and perform local adoption by fine-tuning in each client.

## 6.2 Federated Graph Learning

Due to the great prowess of FL, it is natural to apply FL to graph data and solve the data isolation issue. Recently, a cornucopia of studies has extended FL to graph data for different downstream tasks, such as node classification [48], knowledge graph completion [4], and graph classification [38, 47], cross-client missing information completion [29, 51]. Compared with generic FL, node attributes and graph structures get entangled simultaneously in the data heterogeneity issue of FGL. To handle this issue, a handful of studies proposed their approaches. For example, GCFL [47] and FedStar [38] are two recent frameworks for graph classification in FGL. The authors of GCFL [47] investigate common and diverse properties in intra- and cross-domain graphs. They employ Clustered FL [30] in GCFL to encourage clients with similar properties to share model parameters. A following work FedStar [38] aims to jointly train a global structure encoder in the feature-structure decoupled GNN across clients. FedLit [48] mitigates the impact of link-type heterogeneity underlying homogeneous graphs in FGL via an EM-based clustering algorithm.

# 7 CONCLUSION

In this study, we investigate the problem of divergent neighboring information in FGL. With the high node heterophily, minority nodes in a client can aggregate adverse neighboring information in GNN models and obtain biased node embeddings. To grapple with this issue, we propose FedSpray, a novel FGL framework that aims to learn personalized GNN models for each client. FedSpray extracts and shares class-wise structure proxies learned by a global feature-structure encoder. The structure proxies serve as unbiased neighboring information to obtain soft targets generated by the feature-structure encoder. Then, FedSpray uses the soft labels to regularize local training of the GNN models and, therefore, eliminate the impact of adverse neighboring information on node embeddings. We conduct extensive experiments over four real-world datasets to validate the effectiveness of FedSpray. The experimental results demonstrate the superiority of our proposed FedSpray compared with the state-of-the-art baselines.

# ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation under grants IIS-2006844, IIS-2144209, IIS-2223769, IIS-2331315, CNS-2154962, and BCS-2228534; the Commonwealth Cyber Initiative Awards under grants VV-1Q23-007, HV-2Q23-003, and VV-1Q24-011; the JP Morgan Chase Faculty Research Award; the Cisco Faculty Research Award; and Snap gift funding.

# REFERENCES

[1] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* (2008).

[2] Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji. 2021. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[3] Hong-You Chen and Wei-Lun Chao. 2022. On Bridging Generic and Personalized Federated Learning for Image Classification. In *International Conference on Learning Representations*.

[4] Mingyang Chen, Wen Zhang, Zonggang Yuan, Yantao Jia, and Huajun Chen. 2021. Fede: Embedding knowledge graphs in federated setting. In *The 10th International Joint Conference on Knowledge Graphs*.

[5] Daniel Daza, Michael Cochez, and Paul Groth. 2021. Inductive entity representations from text via link prediction. In *Proceedings of the Web Conference 2021*.

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

[7] Yushun Dong, Binchi Zhang, Yiling Yuan, Na Zou, Qi Wang, and Jundong Li. 2023. Reliant: Fair knowledge distillation for graph neural networks. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*.

[8] Santo Fortunato and Darko Hric. 2016. Community detection in networks: A user guide. *Physics reports* 659 (2016), 1–44.

[9] Xingbo Fu, Chen Chen, Yushun Dong, Anil Vullikanti, Eili Klein, Gregory Madden, and Jundong Li. 2023. Spatial-Temporal Networks for Antibiogram Pattern Prediction. In *2023 IEEE 11th International Conference on Healthcare Informatics (ICHI)*.

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

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*.

[12] Yilin He, Chaojie Wang, Hao Zhang, Bo Chen, and Mingyuan Zhou. 2022. A variational edge partition model for supervised graph representation learning. *Advances in Neural Information Processing Systems* 35 (2022), 12339–12351.

[13] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[14] Wenke Huang, Guancheng Wan, Mang Ye, and Bo Du. 2023. Federated graph semantic and structural learning. In *Proc. Int. Joint Conf. Artif. Intell.*

[15] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*.

[16] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

[17] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

[18] Qinbin Li, Bingsheng He, and Dawn Song. 2021. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[19] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. 2021. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*.

[20] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. In *Proceedings of Machine learning and systems*.

[21] Quande Liu, Cheng Chen, Jing Qin, Qi Dou, and Pheng-Ann Heng. 2021. Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[22] Zhiwei Liu, Liangwei Yang, Ziwei Fan, Hao Peng, and Philip S Yu. 2021. Federated social recommendation with graph neural network. *ACM Transactions on Intelligent Systems and Technology* (2021).

[23] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. 2022. Revisiting heterophily for graph neural networks. *Advances in neural information processing systems* (2022).

[24] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. 2022. Is Homophily a Necessity for Graph Neural Networks?. In *International Conference on Learning Representations*.

[25] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*.

[26] Péter Mernyei and Cătălina Cangea. 2020. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901* (2020).

[27] Xutong Mu, Yulong Shen, Ke Cheng, Xueli Geng, Jiaxuan Fu, Tao Zhang, and Zhiwei Zhang. 2023. Fedproc: Prototypical contrastive federated learning on non-iid data. *Future Generation Computer Systems* (2023).

[28] Jaehoon Oh, Sangmook Kim, and Se-Young Yun. 2022. Fedbabu: Towards enhanced representation for federated image classification. In *International Conference on Learning Representations*.

[29] Liang Peng, Nan Wang, Nicha Dvornek, Xiaofeng Zhu, and Xiaoxiao Li. 2022. Fedni: Federated graph learning with network inpainting for population-based disease prediction. *IEEE Transactions on Medical Imaging* (2022).

[30] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2020. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[31] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* (2008).

[32] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. 2021. Personalized federated learning using hypernetworks. In *International Conference on Machine Learning*.

[33] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[34] Yujun Shi, Jian Liang, Wenqing Zhang, Vincent YF Tan, and Song Bai. 2023. Towards Understanding and Mitigating Dimensional Collapse in Heterogeneous Federated Learning. In *International Conference on Learning Representations*.

[35] Jaeyun Song, Joonhyung Park, and Eunho Yang. 2022. TAM: topology-aware margin loss for class-imbalanced node classification. In *International Conference on Machine Learning*.

[36] Dianbo Sui, Yubo Chen, Jun Zhao, Yantao Jia, Yuantao Xie, and Weijian Sun. 2020. Feded: Federated learning via ensemble distillation for medical relation extraction. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*.

[37] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2022. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[38] Yue Tan, Yixin Liu, Guodong Long, Jing Jiang, Qinghua Lu, and Chengqi Zhang. 2023. Federated learning on non-iid graphs via structural knowledge sharing. In *Proceedings of the AAAI conference on artificial intelligence*.

[39] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. 2022. Fedproto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[40] Anton Tsitsulin, Benedek Rozemberczki, John Palowitch, and Bryan Perozzi. 2022. Synthetic graph generation to benchmark graph learning. *arXiv preprint arXiv:2204.01376* (2022).

[41] Song Wang, Yushun Dong, Binchi Zhang, Zihan Chen, Xingbo Fu, Yinhan He, Cong Shen, Chuxu Zhang, Nitesh V Chawla, and Jundong Li. 2024. Safety in Graph Machine Learning: Threats and Safeguards. *arXiv preprint arXiv:2405.11034* (2024).

[42] Song Wang, Xingbo Fu, Kaize Ding, Chen Chen, Huiyuan Chen, and Jundong Li. 2023. Federated few-shot learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[43] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. Fedgnn: Federated graph neural network for privacy-preserving recommendation. In *International Conference on Machine Learning Workshops*.

[44] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*.

[45] Yebo Wu, Li Li, Chunlin Tian, and Chengzhong Xu. 2024. Breaking the Memory Wall for Heterogeneous Federated Learning with Progressive Training. *arXiv preprint arXiv:2404.13349* (2024).

[46] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* (2020).

[47] Han Xie, Jing Ma, Li Xiong, and Carl Yang. 2021. Federated graph classification over non-iid graphs. In *Advances in neural information processing systems*.

[48] Han Xie, Li Xiong, and Carl Yang. 2023. Federated node classification over graphs with latent link-type heterogeneity. In *Proceedings of the ACM Web Conference 2023*.

[49] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. 2022. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *IEEE International Conference on Data Mining*.

[50] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2020. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*.

[51] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. 2021. Subgraph federated learning with missing neighbor generation. In *Advances in neural information processing systems*.

[52] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. 2022. Graph-less Neural Networks: Teaching Old MLPs New Tricks Via Distillation. In *International Conference on Learning Representations*.

# A PROOF OF PROPOSITION 3.1

Proposition 3.1. *Given a set of $K$ clients, each client $k$ owns a local graph $\mathcal{G}^{(k)} \sim Gen(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, p^{(k)}, q^{(k)})$, $dist = \frac{||\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2||_2}{2}$, which is smaller than $dist' = \left(1 + \sum_{k=1}^{K}(1 - q^{(k)})(p^{(k)} - \frac{1}{2})\right)\frac{||\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2||_2}{2}$.*

Proof. Without loss of generality, we assume that the majority class is $c_1$ for each client $k = 1, 2, \cdots, M$ and $c_2$ for each client $k = M+1, M+2, \cdots, K$. Based on the neighborhood distributions, the neighboring features aggregated by the message-passing mechanism in GNNs follow Gaussian distribution

$$\mathbf{h}_i^{(k)} \sim N\left(p^{(k)}\boldsymbol{\mu}_1 + (1 - p^{(k)})\boldsymbol{\mu}_2, \frac{\mathbf{I}}{\sqrt{|\mathcal{N}(v_i)|}}\right) \tag{16}$$

for each client $k = 1, 2, \cdots, M$, and

$$\mathbf{h}_i^{(k)} \sim N\left((1 - p^{(k)})\boldsymbol{\mu}_1 + p^{(k)}\boldsymbol{\mu}_2, \frac{\mathbf{I}}{\sqrt{|\mathcal{N}(v_i)|}}\right) \tag{17}$$

for each client $k = M+1, M+2, \cdots, K$.

The expectation of node embeddings after the message-passing mechanism will be $\mathbb{E}_{c_1}[\mathbf{x}_i^{(k)} + \mathbf{h}_i^{(k)}]$ for class $c_1$ and $\mathbb{E}_{c_2}[\mathbf{x}_i^{(k)} + \mathbf{h}_i^{(k)}]$ for class $c_2$. We omit the linear transformation because it can be absorbed in the linear GNN classifiers. The decision boundary of the optimal linear classifier is defined by the hyperplane $\mathcal{P}$ that is orthogonal to

$$\begin{aligned}
&\mathbb{E}_{c_1}[\mathbf{x}_i^{(k)} + \mathbf{h}_i^{(k)}] - \mathbb{E}_{c_2}[\mathbf{x}_i^{(k)} + \mathbf{h}_i^{(k)}] \\
=& \mathbb{E}_{c_1}[\mathbf{x}_i^{(k)}] + \mathbb{E}_{c_1}[\mathbf{h}_i^{(k)}] - \mathbb{E}_{c_2}[\mathbf{x}_i^{(k)}] - \mathbb{E}_{c_2}[\mathbf{h}_i^{(k)}]
\end{aligned} \tag{18}$$

For each client $k$, we have $\mathbb{E}_{c_1}[\mathbf{h}_i^{(k)}] = \mathbb{E}_{c_2}[\mathbf{h}_i^{(k)}]$. Therefore,

$$\begin{aligned}
&\mathbb{E}_{c_1}[\mathbf{x}_i^{(k)} + \mathbf{h}_i^{(k)}] - \mathbb{E}_{c_2}[\mathbf{x}_i^{(k)} + \mathbf{h}_i^{(k)}] \\
=& \mathbb{E}_{c_1}[\mathbf{x}_i^{(k)}] - \mathbb{E}_{c_2}[\mathbf{x}_i^{(k)}] = \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2,
\end{aligned} \tag{19}$$

and the distance from each class to $\mathcal{P}$ is

$$dist = \frac{||\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2||_2}{2}. \tag{20}$$

Let the server collect neighboring information from each client via FedAvg. The global neighboring information will be

$$\mathbf{s}_1 = \sum_{k=1}^{M}\mathbf{h}_i^{(k)} + \sum_{k=M+1}^{K} q^{(k)}\mathbf{h}_i^{(k)} \tag{21}$$

for class 1 and

$$\mathbf{s}_2 = \sum_{k=1}^{M} q^{(k)}\mathbf{h}_i^{(k)} + \sum_{k=M+1}^{K}\mathbf{h}_i^{(k)} \tag{22}$$

for class 2. In this case, we replace $\mathbf{h}_i^{(k)}$ in Eq. (19) and get the new hyperplane $\mathcal{P}'$ that is orthogonal to

$$\begin{aligned}
&\mathbb{E}_{c_1}[\mathbf{x}_i^{(k)} + \mathbf{s}_1] - \mathbb{E}_{c_2}[\mathbf{x}_i^{(k)} + \mathbf{s}_2] \\
=& \mathbb{E}_{c_1}[\mathbf{x}_i^{(k)}] + \mathbb{E}_{c_1}[\mathbf{s}_1] - \mathbb{E}_{c_2}[\mathbf{x}_i^{(k)}] - \mathbb{E}_{c_2}[\mathbf{s}_2] \\
=& \mathbb{E}_{c_1}[\mathbf{x}_i^{(k)}] - \mathbb{E}_{c_2}[\mathbf{x}_i^{(k)}] + \mathbb{E}_{c_1}[\mathbf{s}_1] - \mathbb{E}_{c_2}[\mathbf{s}_2] \\
=& \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 + \mathbb{E}_{c_1}[\mathbf{s}_1] - \mathbb{E}_{c_2}[\mathbf{s}_2],
\end{aligned} \tag{23}$$

where

$$\begin{aligned}
&\mathbb{E}_{c_1}[\mathbf{s}_1] - \mathbb{E}_{c_2}[\mathbf{s}_2] \\
=& \mathbb{E}_{c_1}\left[\sum_{k=1}^{M}\mathbf{h}_i^{(k)} + \sum_{k=M+1}^{K} q^{(k)}\mathbf{h}_i^{(k)}\right] \\
&- \mathbb{E}_{c_2}\left[\sum_{k=1}^{M} q^{(k)}\mathbf{h}_i^{(k)} + \sum_{k=M+1}^{K}\mathbf{h}_i^{(k)}\right] \\
=& \sum_{k=1}^{M}\mathbb{E}_{c_1}[\mathbf{h}_i^{(k)}] + \sum_{k=M+1}^{K} q^{(k)}\mathbb{E}_{c_1}[\mathbf{h}_i^{(k)}] \\
&- \sum_{k=1}^{M} q^{(k)}\mathbb{E}_{c_2}[\mathbf{h}_i^{(k)}] - \sum_{k=M+1}^{K}\mathbb{E}_{c_2}[\mathbf{h}_i^{(k)}] \\
=& \sum_{k=1}^{M}(1 - q^{(k)})(p^{(k)}\boldsymbol{\mu}_1 + (1 - p^{(k)})\boldsymbol{\mu}_2) \\
&+ \sum_{k=M+1}^{K}(q^{(k)} - 1)((1 - p^{(k)})\boldsymbol{\mu}_1 + p^{(k)}\boldsymbol{\mu}_2) \\
=& \left(\sum_{k=1}^{K}(1 - q^{(k)})p^{(k)} - \sum_{k=M+1}^{K}(1 - q^{(k)})\right)\boldsymbol{\mu}_1 \\
&+ \left(\sum_{k=1}^{K}(q^{(k)} - 1)p^{(k)} - \sum_{k=1}^{M}(q^{(k)} - 1)\right)\boldsymbol{\mu}_2 \\
=& \sum_{k=1}^{K}(1 - q^{(k)})p^{(k)}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\
&+ \sum_{k=1}^{M}(1 - q^{(k)})\boldsymbol{\mu}_2 - \sum_{k=M+1}^{K}(1 - q^{(k)})\boldsymbol{\mu}_1 \\
=& \sum_{k=1}^{K}(1 - q^{(k)})(p^{(k)} - \frac{1}{2})(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\
&+ (\frac{\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2}{2})\left(\sum_{k=1}^{M}(1 - q^{(k)}) - \sum_{k=M+1}^{K}(1 - q^{(k)})\right).
\end{aligned}$$

Given the balanced global distribution where we have the same number of nodes from class $c_1$ and $c_2$, $\sum_{k=1}^{M}(1 - q^{(k)}) - \sum_{k=M+1}^{K}(1 - q^{(k)})$ in the second term will be equal to 0. Therefore, the above equation can be simplified as

$$\mathbb{E}_{c_1}[\mathbf{s}_1] - \mathbb{E}_{c_2}[\mathbf{s}_2] = \sum_{k=1}^{K}(1 - q^{(k)})(p^{(k)} - \frac{1}{2})(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2). \tag{24}$$

Then the new hyperplane $\mathcal{P}'$ is orthogonal to

$$\left(1 + \sum_{k=1}^{K}(1 - q^{(k)})(p^{(k)} - \frac{1}{2})\right)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2), \tag{25}$$

which is in the same direction of $\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$. Given $0 < q^{(k)} < 1$ and $\frac{1}{2} < p^{(k)} < 1$ for each client $k$, the distance from each class to $\mathcal{P}'$ is

$$dist' = \left(1 + \sum_{k=1}^{K}(1 - q^{(k)})(p^{(k)} - \frac{1}{2})\right)\frac{||\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2||_2}{2}, \tag{26}$$

which completes the proof. □

**Table 4: The statistics and basic information about the four datasets adopted for our experiments.**

| Dataset | PubMed | WikiCS | Physics | Flickr |
|---|---|---|---|---|
| **Clients** | 7 | 12 | 12 | 20 |
| **Node Features** | 500 | 300 | 8,415 | 500 |
| **Average Nodes** | 1,608 | 861 | 2,651 | 4,441 |
| **Average Edges** | 3,600 | 11,721 | 14,790 | 14,331 |
| **Classes** | 3 | 10 | 5 | 7 |

**Table 5: Classification accuracy of FedSpray and Fedavg on node classification over WikiCS with GCN.**

| Epochs | Node sets | FedAvg | FedSpray |
|---|---|---|---|
| $E = 3$ | Overall | 80.38 ± 0.85 | 81.37 ± 0.39 |
| | Minority | 35.04 ± 2.79 | 47.43 ± 1.39 |
| $E = 5$ | Overall | 80.53 ± 0.74 | 81.51 ± 0.45 |
| | Minority | 35.48 ± 2.52 | 47.43 ± 1.31 |
| $E = 10$ | Overall | 80.23 ± 0.70 | 81.43 ± 0.51 |
| | Minority | 35.13 ± 2.47 | 46.85 ± 1.18 |

# B EXPERIMENT DETAILS

## B.1 Datasets

Here we provide a detailed description of the four datasets we adopted to support our argument. These datasets are commonly used in graph learning from various domains: PubMed in citation network, WikiCS in web knowledge, Physics in co-author graph, and Flickr in social images. Table 4 summarizes the statistics and basic information of the distributed graph data.

## B.2 Baselines

We compare our FedSpray with six baselines in our experiments. We provide the details of these baselines as follows.

- **Local**: Models are locally trained on each client using its local data, without any communication with the server or other clients for collaborative training.
- **FedAvg** [25]: It is a foundation method of FL that operates by aggregating local updates from clients and computing a weighted average of the updates to update the global model.
- **APFL** [6]: APFL empowers clients to utilize a combination of local and global models as their personalized model. Additionally, during training, APFL autonomously determines the optimal mixing parameter for each client, ensuring superior generalization performance, even in the absence of prior knowledge regarding the diversity among the data of different clients.
- **GCFL** [47]: GCFL employs a clustering mechanism based on gradient sequences to dynamically group local models using GNN gradients, effectively mitigating heterogeneity in both graph structures and features.
- **FedStar** [38]: FedStar is devised to extract and share structural information among graphs. It accomplishes this through the utilization of structure embeddings and an independent structure encoder, which is shared across clients while preserving personalized feature-based knowledge.
- **FedLit** [48]: FedLit is an FL framework tailored for graphs with latent link-type heterogeneity. It employs a clustering algorithm to dynamically identify latent link types and utilizes multiple convolution channels to adapt message-passing according to these distinct link types.

## B.3 Extra Experimental Results

*B.3.1* ***Results with Varying Local Epochs.*** In FL, clients usually perform multiple local training epochs before global aggregation to reduce communication costs. We show the results of FedSpray and FedAvg with varying local epochs in Table 5. The results demonstrate that FedSpray can consistently outperform FedAvg with different local epochs.