FedHERO: A Federated Learning Approach for Node Classification Task on Heterophilic Graphs

Anonymous authors
Paper under double-blind review

Abstract

Graph neural networks (GNNs) have shown significant success in modeling graph data, and Federated Graph Learning (FGL) empowers clients to collaboratively train GNNs in a distributed manner while preserving data privacy. However, FGL faces unique challenges when the general neighbor distribution pattern of nodes varies significantly across clients. Specifically, FGL methods usually require that the graph data owned by all clients is homophilic to ensure similar neighbor distribution patterns of nodes. Such an assumption ensures that the learned knowledge is consistent across the local models from all clients. Therefore, these local models can be properly aggregated as a global model without undermining the overall performance. Nevertheless, when the neighbor distribution patterns of nodes vary across different clients (e.g., when clients hold graphs with different levels of heterophily), their local models may gain different and even conflict knowledge from their node-level predictive tasks. Consequently, aggregating these local models usually leads to catastrophic performance deterioration on the global model. To address this challenge, we propose FedHERO, an FGL framework designed to harness and share insights from heterophilic graphs effectively. At the heart of FedHERO is a dual-channel GNN equipped with a structure learner, engineered to discern the structural knowledge encoded in the local graphs. With this specialized component, FedHERO enables the local model for each client to identify and learn patterns that are universally applicable across graphs with different patterns of node neighbor distributions. FedHERO not only enhances the performance of individual client models by leveraging both local and shared structural insights but also sets a new precedent in this field to effectively handle graph data with various node neighbor distribution patterns. We conduct extensive experiments to validate the superior performance of FedHERO against existing alternatives.

1 Introduction

Graph Neural Network (GNN) aims to extract informative patterns from graphs (Xia et al., 2021; Wu et al., 2020b; Zhou et al., 2020; Scarselli et al., 2008; Xu et al., 2018a). However, graph data may contain sensitive information about the involved individuals (e.g., a user's race in a social network (Dwork et al., 2012; Beutel et al., 2017)). As a consequence, sharing graph data across different stakeholders is raising concerns due to the potential risk of privacy leakage, which further prevents training GNNs in a centralized manner (Cong & Mahdavi, 2022; Dai et al., 2022; Mei et al., 2019; Zhu & Zhu, 2020). To properly handle the above-mentioned problems, Federated Graph Learning (FGL) has emerged as a popular collaborative learning framework, which enables participants (clients) to derive insights from distributed graph sources while upholding stringent privacy safeguards (Liu et al., 2022a; Lou et al., 2021; Zhu et al., 2022; Zhang et al., 2022; Peng et al., 2021; Chen et al., 2021b). A typical scenario of FGL is that each client possesses a single graph (local graph) and seeks to collaboratively train a global GNN model for certain tasks (e.g., node classification tasks (Xiao et al., 2022; Bhagat et al., 2011)) without sharing the raw graph data. In such scenarios, current FGL approaches implicitly rely on the assumption of homophilic local graphs (Zhu et al., 2020; Ma et al., 2021), i.e., nodes within a client's graph are more likely to form connections with other nodes of the same class (Ma et al., 2019; Tang et al., 2009b). Specifically, when clients hold homophilic

graphs, most of the neighbors of each node will have the same class as this center node (Ying et al., 2018). The GNNs trained on different clients will thus capture a similar tendency, and aggregating these GNNs usually makes the global model in FGL better learn such a tendency and benefit the generalization ability of the global model (Baek et al., 2023; Zhang et al., 2021).

Nevertheless, a critical issue regarding the above-mentioned assumption is that it overlooks the prevalence of heterophilic graphs (Pandit et al., 2007; Zheng et al., 2022; 2023; Bo et al., 2021; Yang et al., 2021; Chien et al., 2020; Liu et al., 2023a), where nodes belonging to different classes tend to be connected. For instance, in financial transaction networks, nodes denote bank clients, where node labels are their credit risk and edges represent the financial interactions between them. Such a network can be heterophilic, and we show an example in Figure 1. Specifically, a 'Moderate Risk' customer in Bank A may transact with multiple 'Low Risk' individuals, whereas in Bank B, the 'Moderate Risk' customer engages primarily with 'High Risk' counterparts. When training GNNs to predict customer risk, Bank A's GNN learns the connection between 'Moderate Risk' and 'Low Risk,' while Bank B's GNN captures the connection between 'Moderate Risk' and 'High Risk.' Directly aggregation models could lead to poor performance in predicting 'Moderate Risk' customers, as they fail to establish a consistent pattern that generalizes well across both banks. Consequently, this mismatch can result in performance degradation for FGL algorithms.

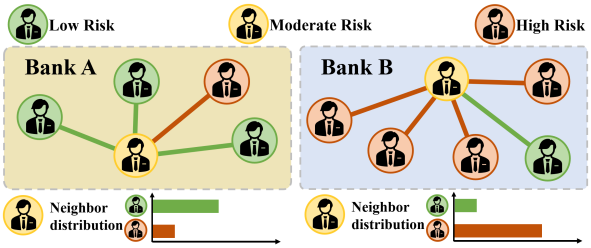


Figure 1: An example of financial transaction networks in two banks. The edges in the figure represent transaction records. Due to the diverse customer financial habits in distinct networks, customers (nodes) within the same risk classification (label) across different banks have diverse transactional relationship patterns (neighbor distribution).

To properly handle the challenge discussed above, in this paper, we introduce FedHERO, a novel framework for **Fed**erated Learning on **He**te**ro**philic Graphs. FedHERO is equipped with a carefully designed dual-channel GNN model, including the global channel and the local channel. The dual-channel design aims to aggregate GNNs that have learned similar patterns to facilitate knowledge sharing. To ensure each client obtains a GNN model aligned with others, we introduce a shared structure learning model that generates latent graphs across clients based on common patterns. By learning from and sharing the knowledge from these latent graphs, GNNs in the global channel capture similar tendencies, enhancing their overall performance. Meanwhile, the local channel operates directly on the original graph. With such a design, FedHERO facilitates the sharing of common knowledge across clients while leveraging the unique structural information in each client's local graph to enhance overall model performance. This approach offers two key advantages: (1) The global channel, trained on the latent graph, reduces reliance on the original neighbor distribution. The GNN models trained on different clients could capture similar tendencies from the shared structure learning model, and aggregating these GNNs could exploit FGL's potential. (2) Retaining the locally trained GNN on the original graph preserves sensitive information, enhancing privacy protection. We summarize the contributions of our paper below.

- **Problem Formulation.** To the best knowledge, we take an initial step to formulate and handle the challenge arising from the implicit heterophilic assumption of FGL.

- **Algorithmic Design.** We introduce an innovative FGL framework, named FedHERO, which empowers clients to acquire global structure learning models while preserving graph-specific structural insights at the local level.

- **Empirical Evaluation.** We perform comprehensive experiments across four real-world graph datasets and three synthetic datasets. The results demonstrate that FedHERO consistently outperforms existing FGL alternatives significantly.

2 PRELIMINARIES

Notations. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph containing N nodes $\mathcal{V}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set. We define $\mathbf{X} \in \mathbb{R}^{N \times d}$ as the feature matrix, where $\mathbf{x}_v \in \mathbb{R}^d$ represents the attributes of node v. The initial graph structure can be represented in an adjacency matrix $\mathbf{A} = \{a_{uv}\}_{N \times N}$, where $a_{uv} = 1$ indicates the presence of an edge between node u and v and 0 otherwise. $\mathbf{Y} \in \mathbb{R}^{N \times C}$ denotes the label matrix, where $\mathbf{y}_v$ represents the label vector for node v, and C is the number of classes. GNNs operate by representing nodes based on information from both their neighborhoods and their own attributes, which can be expressed as follows:

$$\mathbf{z}_v^l = \mathrm{UPD}(\mathbf{z}_v^{l-1}, \mathrm{AGG}(\{\mathbf{z}_u^{l-1} : \forall u \text{ s.t. } a_{uv} = 1\})), \tag{1}$$

where $\mathbf{z}_v^l$ is the embedding of the node v at l-th layer. The operation AGG aggregates the embeddings of v's neighbors, while UPD updates the representation of node v using its embedding from the previous layer and the aggregated embeddings from its neighbors. Initially, $\mathbf{z}_v^0$ is set as $\mathbf{x}_v$.

Federated Learning. The objective of clients in FL is to jointly train models using local data. Let's consider M data owners, each possessing data exclusively accessible to them, denoted as $\mathcal{D}_i = (\mathcal{X}_i, \mathcal{Y}_i)$, where $\mathcal{X}_i$ represents the data instances and $\mathcal{Y}_i$ is the corresponding labels. N_i signifies the number of data samples held by client i and $N = \sum_i N_i$ denotes the total number of data samples available. $\mathcal{L}_i$ and w_i are the loss function and model parameters of client i, respectively. The FL process typically encompasses the following steps: (i) At r-th round, the server selects a subset of clients that participate in the training process. These clients receive the global model from the last round $\overline{w}_r$ and initialize the local model parameters as $w_i^r \leftarrow \overline{w}_r$. (ii) Each selected client updates its local model using its own local data to minimize the task loss: $w_i^{r+1} \leftarrow w_i^r - \eta \nabla \mathcal{L}_i(w_i^r)$. (iii) The server aggregates the updated local models to obtain an updated global model: $\overline{w}_{r+1} = \sum_i \frac{N_i}{N} w_i^{r+1}$. This process continues iteratively until convergence is achieved.

Preliminary Study. To highlight the unique challenges posed by varying neighbor distribution patterns across clients, we conduct a preliminary study comparing the performance of local training models with the global model produced by FedAvg. Table 1 contrasts results on heterophilic datasets (Rozemberczki et al., 2021) (Squirrel and Chameleon) with those on homophilic datasets (PubMed (Namata et al., 2012) and Citeseer (Sen et al., 2008)). On homophilic graphs, the global model generally shows better performance. In contrast, for heterophilic graphs, the aggregated global model underperforms relative to local training models. This discrepancy arises because clients with heterophilic graphs are more likely to exhibit diverse neighbor distribution patterns, leading locally trained GNNs to capture varied tendencies. As a result, the global model struggles to aggregate and generalize this knowledge, resulting in poor predictions. This preliminary experiment underscores the impact of distinct neighbor distribution patterns on FGL performance, reinforcing the need to address the challenges identified in our study.

Table 1: Performances of the locally trained model and FedAvg's global model on four datasets.

Datasets	**Squirrel**	**Chameleon**	**PubMed**	**Citeseer**
	heterophilic		homophilic	
Local	32.92±1.94	47.45±0.49	84.47±0.47	67.89±0.54
FedAvg	28.55±0.45	38.65±2.19	87.08±0.01	72.41±0.22

Problem Setup. In the FGL system, we assume that M clients hold M graphs and aim to collaboratively train node classification models. In particular, client i owns the graph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, and $\mathbf{A}_i$ denotes the corresponding adjacency matrix of graph $\mathcal{G}_i$.

We aim to train a universal structure learner g_L parameterized by θ across local graphs. This learner should capture the universally applicable patterns across graphs $\{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_M\}$. The trained structure learner is expected to generate effective graph structures that enhance the downstream classification performance of local GNN classifiers. Formally, the goal of training g_L alongside the local models f_i can be expressed as an optimization problem:

$$\min_{\theta, w_1, \ldots, w_M} \sum_{i=1}^{M} \mathcal{L}_i(f_i(g_L(\mathbf{X}_i, \mathbf{A}_i), \mathbf{X}_i, \mathbf{A}_i), \mathbf{Y}_i), \tag{2}$$

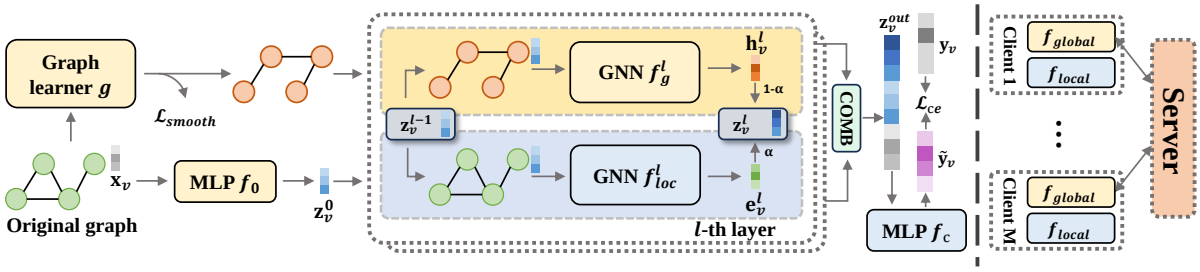


Figure 2: The framework of FEDHERO. Blue boxes represent models in the f_{local} channel, which are trained locally and personalized for each client. Yellow boxes denote models in the f_{global} channel, shared among clients to train a structure learner for mutual benefit. On the right side of the figure, the model aggregation scheme in FEDHERO is depicted.

where g_L takes the node feature $\mathbf{X}_i$ and adjacency matrix $\mathbf{A}_i$ of the local graph as input to generate a new graph structure, and the local model f_i predicts the node classes from the new model structure and the original graph data.

3 Proposed Framework: FedHERO

We aim to address the issue of distinct neighbor distribution patterns, particularly for heterophilic graphs. In this context, clients possess heterophilic graphs in which the neighbor distribution patterns of nodes vary across different clients. As discussed in Section 1, this variability can result in diverse aggregation effects. Models trained on graphs with distinct neighbor distributions exhibit limited generalization ability.

To address this challenge, we present FEDHERO, as illustrated in Figure 2. First, we outline the implementation of structure learning and its initialization. We then demonstrate how FEDHERO integrates the global graph structure pattern and local structural information, detailing the joint optimization process involving structure learning and personalized task models. Lastly, we explore the mechanism for sharing structural knowledge within FEDHERO.

3.1 Graph Structure Learning in FedHERO

Graph structure learning endeavors to comprehend the message-flow pattern throughout the graph (Xia et al., 2021). Nevertheless, in FL scenarios where data is diversely distributed and privacy concerns are paramount, directly aggregating all data to train a comprehensive structure learning model becomes unfeasible. To tackle this challenge, we implement a structure learning model for each client, which can learn structural knowledge from local data. By sharing the learned knowledge with other clients, the entire FGL system can acquire a unified, globally optimized structural representation that captures universally applicable patterns across diverse graphs.

The objective of the structure learning model is to generate relevant latent structures that facilitate effective message passing based on the original graph. A common approach considers representing the weight of an edge between two nodes as a distance measure between two end nodes (Zhao et al., 2021b; Kazi et al., 2022; Chen et al., 2020). We follow this idea and employ a one-layer GNN to derive node representations $\mathbf{Z}$ from the original graph. The structure learning model can be represented by learning a metric function $\phi(\cdot,\cdot)$ of a pair of representations: $\tilde{a}_{uv} = \phi(\mathbf{z}_u, \mathbf{z}_v)$, where $\mathbf{z}_u$ is the learned embedding of node u produced by GNN. The metric function $\phi(\cdot,\cdot)$ can assume various forms like cosine distance (Nguyen & Bai, 2010) and RBF kernel (Li et al., 2018b). In this study, we employ the multi-head attention mechanism as the metric function, following the approach outlined by Chen et al. (2020):

$$\phi(\mathbf{z}_u, \mathbf{z}_v) = \frac{1}{N_H} \sum_{h=1}^{N_H} \cos(\mathbf{w}_h^1 \odot \mathbf{z}_u, \mathbf{w}_h^2 \odot \mathbf{z}_v), \tag{3}$$

where $\cos(\cdot,\cdot)$ is cosine similarity, and $\odot$ denotes the Hadamard product. By consolidating the outputs N_H heads and using $\mathbf{w}_h^1, \mathbf{w}_h^2$ to scale the node embeddings individually, the structure learning model can potentially identify connections between similar or dissimilar nodes and exhibit the necessary expressive power to handle graphs with heterophily. First, aggregating outputs of different heads can enhance the model's expressiveness in capturing the underlying influence between nodes from various perspectives. Additionally, the weight vectors $\mathbf{w}_h^1$ and $\mathbf{w}_h^2$ are introduced to learn element-wise scaling of input vectors, i.e., node representations. Moreover, the use of two independent weight vectors $\mathbf{w}_h^1$ and $\mathbf{w}_h^2$ allows for potential alignment or misalignment, enabling the model to connect both similar and dissimilar nodes. It's important to highlight that FEDHERO can be integrated with various metric functions and even various structure learning methods. Further empirical results are illustrated in Section 4.4.

To construct the latent graph $\tilde{\mathbf{A}} \in \mathbb{R}^{N\times N}$, we implement a post-processing step to create a kNN graph, wherein each node connects to up to k neighbors, i.e.,

$$\tilde{\mathbf{A}}_{ij} = \begin{cases} 1, & \text{if } \tilde{a}_{ij} \text{ is a top-}k \text{ element in } \{\tilde{a}_{ij} | j \in [n]\}, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

This design choice is based on the differentiability of the top-k function, which facilitates the computation of parameter gradients and subsequent model updates in FEDHERO. Additionally, we explore an alternative method for latent graph generation. Such a method treats each edge in the latent graph as a Bernoulli random variable. Here, each latent edge $\tilde{\mathbf{A}}_{ij}$ is sampled from a Bernoulli distribution with parameter $\tilde{a}_{ij}$ (Wu et al., 2020a; Elinas et al., 2020; Zhao et al., 2023). A comparison of these methods and the impact of varying k values is provided in Appendix C.

3.2 Optimization of the Dual-channel Model

In existing FGL frameworks (Zhang et al., 2021; Baek et al., 2023), the local models primarily employ one or multiple GNNs in sequence. These models initiate the process of generating node representations directly from raw features. However, as Tan et al. (2023) pointed out, directly constructing and sharing these local models across diverse clients can significantly compromise personalized performance due to feature and representation space misalignment. To address this challenge, divide the local model into two channels: f_{global} and f_{local}. The global learner f_{global} takes the original input graph and refines the latent graph structure. Through cooperation with other clients, f_{global} detects patterns of universal relevance across heterophilic graphs. On the other hand, the local task learner f_{local} extracts valuable client-specific information from raw node features and structures within the local graph. This information includes biased structures or diverse feature distributions tailored for follow-up tasks.

In a client's local graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we initiate the process by utilizing a Multi-Layer Perception (MLP) f_0 to extract information from the node feature matrix $\mathbf{X}$ into node embeddings $\mathbf{Z}^0$. Next, we feed the node embeddings $\mathbf{Z}^0$, the latent graph $\tilde{\mathbf{A}}$, and the original graph structure $\mathbf{A}$ into an L-layer GNN. The middle part of Figure 2 depicts how FEDHERO leverages information from both the local graph structure $\mathbf{A}$ and the structure $\tilde{\mathbf{A}}$ from the structure learner to obtain node representations. In the f_{global} channel (models in yellow boxes in Figure 2), the l-th layer of the GNN (denoted as f_g^l) utilizes the node embeddings from the previous layer $\mathbf{Z}^{l-1}$ and the latent graph $\tilde{\mathbf{A}}$ to generate hidden node embeddings that reveal patterns that hold universally across different graphs, denoted as $\mathbf{H}^l$. While feature propagation on the $\tilde{\mathbf{A}}$ could potentially yield enhanced node embeddings that align with global patterns, it is important to note that the original graph structures also contain valuable information, such as effective inductive bias (Bronstein et al., 2017). Therefore, we leverage the information within $\mathbf{A}$. It is combined with the output from the last layer, $\mathbf{Z}^{l-1}$, and fed into the l-th layer of the GNN (denoted as f_{loc}^l) within the f_{local} channel (models in blue boxes in Figure 2). This process generates hidden node embeddings that adhere to the message-passing patterns within the local graph, resulting in $\mathbf{E}^l$. These embeddings play a crucial role in computing the layer-wise updates for node representations:

$$\mathbf{Z}^l = \sigma\Big(\alpha\mathbf{E}^l + (1-\alpha)\mathbf{H}^l\Big), \tag{5}$$

where σ is an activation function and α is a trading hyper-parameter that controls the concentration weight on input structures. Additionally, integrating the outputs from both the f_{global} and f_{local} channels can

enhance training stability. This is achieved by mitigating the influence of substantial variations in latent structures encountered during the training process within the f_{global} channel.

Recent studies show that merging intermediate node representations from multi-hops enables the GNN to integrate nodes of the same class (Zhu et al., 2020; Abu-El-Haija et al., 2019). In addition, encoding the features of the ego node separately from the aggregated neighbor representations enhances node embedding learning (Zhu et al., 2020; Suresh et al., 2021; Yan et al., 2022). Thus, we merge the intermediate node representations generated by each layer of the dual-channel GNNs with the ego feature $\mathbf{X}$: $\mathbf{Z}^{out} = \text{COMB}(\mathbf{X}, \mathbf{Z}^0, ..., \mathbf{Z}^L)$. The COMB function can be implemented in several ways, including max-pooling, concatenation, or LSTM-attention (Xu et al., 2018b). In our approach, we specifically use *concatenation* for the COMB function. This choice is based on the observation that concatenation consistently outperforms other methods in similar contexts (Xu et al., 2018b; Zhu et al., 2020).

In the classification stage, the prediction of the node class is based on its final embedding:

$$\tilde{\mathbf{Y}} = \text{Softmax}(f_c(\mathbf{Z}^{out})), \tag{6}$$

where the classify layer f_c gives the C dimension prediction.

Alongside the classification loss, we incorporate a regularization term to constrain $\tilde{\mathbf{A}}$:

$$\mathcal{L}_{smooth} = \lambda \sum_{u,v} \tilde{\mathbf{A}}_{uv} ||\mathbf{x}_u - \mathbf{x}_v||_2^2 + \mu ||\tilde{\mathbf{A}}||_F^2, \tag{7}$$

where $|| \cdot ||_F$ is the Frobenius norm. The first term in Eq. (7) evaluates the smoothness of the latent graph, operating under the assumption that nodes with smoother features are more likely to be connected. The second term in Eq. (7) serves the purpose of preventing excessively large node degrees. The hyperparameters λ and μ govern the magnitude of the regularization effects. Hence, in conjunction with the cross-entropy loss $\mathcal{L}_{ce}$, we fine-tune the local model using the following loss function:

$$\mathcal{L}_{total} = \mathcal{L}_{ce}(\tilde{\mathbf{Y}}, \mathbf{Y}) + \mathcal{L}_{smooth}. \tag{8}$$

Minimizing the loss function in Eq. (8), the generated latent structure is leveraged to generate node embeddings, providing supplementary information for representation learning.

3.3 Model Aggregation in FedHERO

The right side of Figure 2 illustrates the process of clients sharing local models and receiving updated models from the server. The model aggregation process in FedHERO resembles that of FedAvg (McMahan et al., 2017). However, in the FedHERO framework, the server ***only*** receives and aggregates models from the f_{global} channel of clients instead of the whole model. This sharing mechanism offers two key advantages for FedHERO. First, the structure learning model in the global channel generates latent graphs from node embeddings with relatively low dependence on the original graph structure. By sharing this model among clients, the global channel GNNs learn similar patterns from the generated latent graphs. Aggregating these aligned models enhances the effectiveness of FGL, leading to improved overall performance. Second, retaining the model f_{local}—which focuses on learning the message passing pattern within the local graph—as private and training it locally allows the local model to capture client-specific information, ultimately contributing to subsequent tasks. Let w_g^i denote the parameters of the f_{global} in client i. The server does the aggregation: $\overline{w}_g = \sum_i \frac{N_i}{N} w_g^i$ where N_i represents the number of nodes in the client i's graph, and N denotes the total number of nodes in the graph data of all clients. Subsequently, the server transmits $\overline{w}_g$ to the clients. The clients then update model in the f_{global} channel with $\overline{w}_g$ and commence local training for the ensuing round.

In FedHERO, the information in the f_{local} channel, including the final classifier layer f_c, is not shared. This separation distinguishes between global and personalized information within the local graph. In Section 4.5, we evaluated the impact of sharing different parts of the model. This empirical examination underscored the efficacy of maintaining the f_{local} channel in private, further emphasizing the effectiveness of bifurcating the GNN into two distinct channels.

3.4 Computational and Communication Overhead

We analyze the additional computational and communication overhead introduced by the dual-channel design in FedHERO and compare with current FGL methods, providing insights into its scalability and efficiency in various FL environments.

Let M denote the total number of clients, E represent the number of local training epochs, and L signify the number of layers in the model architecture. For simplicity, let d denote the larger dimension between the node feature dimension d_x and the embedding feature dimension d_{hidden}. $n_{\text{link}}, \theta^d, d_{\text{dg}}$, and d_{rw} regard method-specific parameters. In Table 2, we compute the additional overhead caused by the FGL algorithms compared to FedAvg (McMahan et al., 2017).

Regarding communication overhead, since a relatively small N_H (e.g. $N_H = 4$) enables the structure learner to perform effectively, the additional cost is comparable with other FGL methods like FedLit. As for computational overhead, the primary increase in computational load arises from dual-channel GNN. From Table 2, we can observe that the computation cost of FedHERO aligns with the FGL methods that scale ***linearly*** with the size of node and edge sets. Therefore, FedHERO's design allows it to handle large-scale datasets effectively without introducing much computation burden.

Table 2: Additional computational and communication over- head of FGL methods compared with FedAvg, calculated per client per round.

Methods	Computation cost	Communication overhead
GCFL (Xie et al., 2021)	$\mathcal{O}((M(Ld^2)^2 + MlogM)/E)$	$\mathcal{O}(1)$
FedLit (Xie et al., 2023)	$\mathcal{O}(2\lvert\mathcal{E}\rvert n_{link}dI + \lvert\mathcal{V}\rvert d^2 + 2n_{link}^2 dI/E)$	$\mathcal{O}((n_{link} - 1)Ld^2)$
FedPub (Baek et al., 2023)	$\mathcal{O}(Ld^2 + Md/E)$	$\mathcal{O}(1)$
FedStar (Tan et al., 2023)	$\mathcal{O}(2Ld(d\lvert\mathcal{V}\rvert + \lvert\mathcal{E}\rvert))$	$\mathcal{O}(1)$
FedSage (Zhang et al., 2021)	$\mathcal{O}(\lvert\mathcal{V}\rvert(\theta^d + Ld^2))$	$\mathcal{O}(\lvert\mathcal{V}\rvert d + Ld^2)$
FedHERO	$\mathcal{O}(Ld(d\lvert\mathcal{V}\rvert + \lvert\mathcal{E}\rvert) + N_H d\lvert\mathcal{V}\rvert)$	$\mathcal{O}(N_H d + d^2)$

4 EXPERIMENTS

4.1 Experimental Settings

Datasets. We carry out experiments using well-known heterophily datasets such as Squirrel and Chameleon, derived from Wikipedia datasets (Rozemberczki et al., 2021), along with the Actor dataset (Tang et al., 2009a) and the Flickr dataset (Zeng et al., 2019). Given these datasets' inherent incompatibility with the FGL framework, we adapt them into a federated context using two prevalent graph partitioning methods: the METIS algorithm (Karypis & Kumar, 1997) and the Louvain algorithm (Blondel et al., 2008). This approach enables us to create *semi-synthetic* federated graph learning datasets. As detailed in Appendix A, the graph partitioning process preserves the heterophilic properties of the graphs. Additionally, Appendix B demonstrates significant differences in the neighbor distributions of same-class nodes across clients. This highlights the challenges in FGL that we discussed in the Introduction. Following Baek et al. (2023) and Zhang et al. (2021), we assume no overlapping nodes are shared across data owners. For further details on this process and the overlapping scenario, please refer to Appendix A and C, respectively.

We also include three graph datasets: syn-cora (Zhu et al., 2020), ieee-fraud (Howard et al., 2019), and credit (Yeh & Lien, 2009). These datasets naturally lend themselves to subdivision into multiple subgraphs (e.g., distinct subgraphs within the ieee-fraud dataset correspond to different product transaction records), and we denote them as *real-world* datasets. The description of the datasets is summarized in Appendix A.

Hyper-Parameter Settings. The architecture of the GNN models consists of a feature projection layer, two graph convolutional layers, and a classifier layer. We employ Adam optimizer (Kingma & Ba, 2014) with a learning rate of 0.005. We apply a consistent hyperparameter setting in our main results. Specifically, we

Table 3: Performance of FedHERO and baselines on semi-synthetic heterophilic datasets. Bold fonts indicate the best performances among all methods, while underlines denote the next best results across all methods. M denotes the number of clients.

Datasets	Squirrel				Chameleon				Actor				Flickr			
	METIS			Louvain	METIS			Louvain	METIS			Louvain	METIS			Louvain
	M=5	M=7	M=9		M=3	M=5	M=7		M=5	M=7	M=9		M=30	M=40	M=50	
Local	31.32 (±1.21)	31.44 (±0.11)	31.33 (±0.68)	32.92 (±1.94)	43.77 (±1.07)	44.34 (±3.16)	45.15 (±1.37)	47.45 (±0.49)	29.27 (±0.60)	28.75 (±0.50)	28.37 (±0.34)	26.14 (±0.90)	45.03 (±0.13)	44.98 (±0.36)	45.37 (±0.37)	45.02 (±0.26)
FedAvg	29.67 (±1.04)	28.28 (±0.64)	27.18 (±1.37)	28.55 (±0.45)	42.27 (±2.39)	39.08 (±1.15)	45.09 (±3.24)	38.65 (±2.19)	31.90 (±1.02)	31.45 (±0.72)	31.40 (±0.32)	31.14 (±1.58)	42.45 (±0.2)	42.40 (±0.16)	42.30 (±0.39)	42.56 (±0.19)
FedPub	29.52 (±0.86)	29.38 (±1.04)	29.12 (±0.56)	32.87 (±1.94)	42.36 (±1.50)	46.40 (±1.29)	45.71 (±1.69)	47.16 (±2.93)	28.50 (±0.84)	26.92 (±0.99)	26.95 (±1.20)	26.92 (±1.15)	47.47 (±0.76)	46.64 (±1.28)	46.77 (±0.39)	46.99 (±0.85)
FedStar	31.74 (±0.73)	32.60 (±1.06)	33.53 (±0.91)	36.84 (±0.99)	50.60 (±0.84)	50.42 (±1.14)	50.72 (±1.19)	51.32 (±1.26)	28.58 (±0.74)	27.13 (±0.44)	27.18 (±0.40)	27.43 (±1.19)	49.35 (±0.18)	49.15 (±0.62)	49.08 (±0.25)	49.03 (±0.45)
GCFL	31.40 (±2.09)	30.04 (±1.25)	28.98 (±0.13)	30.42 (±2.28)	45.12 (±1.44)	43.47 (±2.01)	40.30 (±2.81)	38.73 (±3.75)	31.39 (±0.94)	31.63 (±1.23)	31.41 (±1.23)	31.63 (±1.29)	47.65 (±0.22)	47.41 (±0.24)	46.88 (±0.50)	47.37 (±0.39)
FedLit	30.01 (±2.46)	28.22 (±2.59)	27.42 (±2.42)	32.76 (±3.56)	40.10 (±4.45)	34.36 (±4.38)	35.97 (±3.80)	36.67 (±5.06)	32.23 (±1.07)	31.74 (±0.82)	32.16 (±1.13)	32.85 (±1.71)	48.18 (±0.59)	48.35 (±0.36)	48.56 (±0.44)	48.49 (±0.52)
FedSage	33.30 (±0.63)	33.07 (±0.88)	33.10 (±1.02)	35.17 (±2.13)	50.93 (±1.77)	48.72 (±2.61)	49.17 (±2.80)	45.51 (±1.86)	33.75 (±1.02)	32.73 (±0.87)	33.30 (±0.89)	33.57 (±1.83)	45.75 (±0.23)	45.45 (±0.28)	44.74 (±0.36)	50.37 (±0.56)
FedHERO	**36.63** (±0.95)	**36.08** (±0.87)	**37.83** (±1.02)	**38.00** (±0.92)	**53.93** (±0.88)	**54.06** (±0.56)	**52.65** (±1.33)	**55.63** (±2.00)	**35.25** (±0.29)	**34.60** (±1.03)	**34.32** (±0.68)	**34.54** (±0.89)	**50.51** (±0.32)	**50.33** (±0.25)	**49.96** (±0.31)	**50.45** (±0.25)

set $\alpha = 0.2$, $\mu = \lambda = 0.1$, $k = 20$, and $N_H = 4$. In the supplementary material, we provide an analysis of hyperparameters, indicating that optimizing these values could lead to further improvements in performance. The training process involves 200 communication rounds, with each local training epoch lasting one iteration. The nodes are randomly distributed into five groups. For each experiment, one group is chosen as the test set, while three of the remaining groups are utilized for training and one for validation. We perform five experiments on each dataset and report the models' average performance. For the ieee-fraud and credit datasets, we present the mean test Area Under the Curve (AUC), accounting for the imbalanced labels. For the remaining datasets, we use mean test accuracy across clients for evaluation. All methodologies are implemented using PyTorch 1.8 and executed on NVIDIA RTX A6000 GPUs. Our code can be found in the supplementary material.

Baselines. We comprehensively compare FedHERO against five baseline methods in federated graph learning. These baselines encompass strategies that focus on (i) promoting collaboration among clients with similar graph distributions (GCFL (Xie et al., 2021) and Fedpub (Baek et al., 2023)), (ii) sharing structural insights (FedStar (Tan et al., 2023)), (iii) producing missing edges between subgraphs (FedSage (Zhang et al., 2021)), and (iv) discerning and modeling message-passing for latent link types (FedLit (Xie et al., 2023)). To ensure the completeness of our experiments, we also include the standard FL method FedAvg (McMahan et al., 2017).

4.2 Evaluation on Semi-synthetic Datasets

We present the node classification results on four semi-synthetic heterophilic datasets. The average test accuracy and its standard deviation are summarized in Table 3. The results indicate that FedHERO consistently outperforms all the baseline methods by a significant margin. This is because the global channel in FedHERO has less dependence on the local graph structure, with the GNN f_{global} being updated on the latent graph. By sharing the structure learning model across clients, FedHERO ensures that the latent graphs generated by different clients exhibit similar neighbor distribution patterns. This allows the aggregation of consistent information from the global channel, ultimately enhancing the performance.

We also observed that FedAvg often underperforms compared to local training in many experiments. This suggests that heterophilic graphs are more likely to exhibit distinct neighbor distribution patterns across them, which can negatively impact FedAvg's performance. Notably, FedPub and GCFL demonstrate superior overall performance compared to FedAvg. Both encourage clients to collaborate with those who have similar model gradients or outputs. During optimization, the GNN learns to capture graph information,

including neighbor distribution patterns. Therefore, similar model gradients or outputs indicate similar neighbor distribution patterns or tendencies captured by the model. As previously discussed, aggregating models with similar tendencies enhances the performance of the aggregated model.

In our investigation of FedHERO's performance with heterophilic data, we also observed significant improvements in datasets characterized by high heterophily, particularly in the Squirrel and Chameleon datasets. These datasets have homophily ratios of 0.22 and 0.25, respectively, as reported in (Mao et al., 2023), with lower ratios indicating greater heterophily. In comparison to Flickr (homophily ratio: 0.32 (Kim & Oh, 2022)), FedHERO's enhancements are more pronounced. For example, under Louvain partitioning, FedHERO demonstrates improvements of 4.31%, 1.16%, and 0.08% on the Chameleon, Squirrel, and Flickr datasets, respectively, when compared to the best-performing baseline. We believe this phenomenon occurs because, as heterophilic ratio increases, the differences in node neighbor distribution patterns across clients' local graphs become more pronounced, posing greater challenges to existing FGL methods.

4.3 Evaluation on Real-world Datasets

Figure 3 provides a comprehensive overview of the evaluation results across real-world datasets in FGL settings. FedHERO consistently outperforms all FGL baselines on ieee-fraud and credit datasets. The similar performance between our method and other FGL baselines on ieee-fraud and credit datasets may be attributed to distinct class features that are easily distinguishable (e.g., a linear regression model can yield satisfactory results) (Howard et al., 2019) or the low dimensionality of features (Yeh & Lien, 2009). The superior performance of FedSage on the syn-cora dataset may be attributed to GraphSage's utilization of a hybrid design, incorporating elements of both ego-embedding and neighbor embedding. Similar findings are also observed in (Zhu et al., 2020).

Notably, FedPub and FedStar face challenges in achieving strong performance on the syn-cora dataset. The syn-cora dataset consists of random graphs where nodes of different classes have the same expected degree, and different graphs share the same degree distribution (Abu-El-Haija et al., 2019; Zhu et al., 2020). The suboptimal performance of FedStar also demonstrates that relying solely on structural information from the adjacency matrix, while overlooking distinctions between nodes connected by an edge, is insufficient for handling heterophilic graphs. Thus, structure embeddings in FedStar don't contain distinguishable information across classes. Sharing the structure knowledge doesn't benefit training GNNs for node classification. On the other hand, FedPub assumes graphs in clients have community structures, and it leverages the community information. However, the assumption does not hold in syn-cora where random graphs are generated independently.

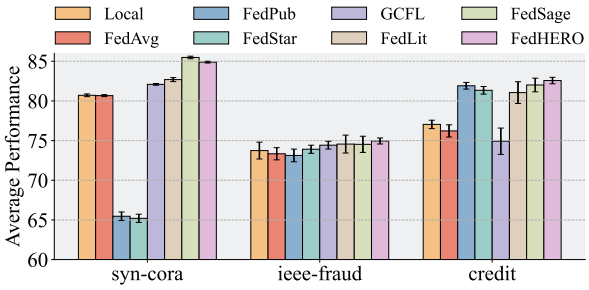


Figure 3: Performance of FedHERO and baselines on three real-world datasets.

4.4 FedHERO with Different Structure Learning Methods

In this section, we explore the adaptability of FedHERO to various structure learning techniques. For example, we substitute the similarity function with an attention-based structure learner, denoted as FedHERO*att*, and with Graph Attention Networks (GAT) (Veličković et al., 2018), denoted as FedHERO*gat*. Another variant, FedHERO*cos*, uses cosine distance as the similarity function. We further examine FedHERO with more complicated graph learning methods, including GLCN (Jiang et al., 2019), GAug (Zhao et al., 2021b). We replace the structure learning model in FedHERO with these methods. The experiments are performed on the Squirrel and Chameleon datasets, and the results are presented in Table 4. From the results, we observe the following:

- The original FedHERO, integrated with multi-head weighted attention, consistently outperforms its variants. This superior performance can be attributed to the ability of multi-head weighted attention to capture a wide range of influences between nodes effectively, making it particularly effective for heterophilic graphs.
- Among basic variants, FedHERO*cos* shows slightly better performance than the other two variants, likely because it avoids additional model parameters, thus reducing the risk of overfitting. Additionally, unlike FedHERO*gat*, FedHERO*cos* considers the entire subgraph for edge generation, which mitigates the impact of heterophilic graphs on the GNN aggregation mechanism (Li et al., 2022; Yang et al., 2022).

Table 4: Performance of FedHERO with different graph structure learning methods.

Variants	**Squirrel**		**Chameleon**	
	METIS M=9	Louvain	METIS M=7	Louvain
FedHERO$_{cos}$	34.65±0.69	37.83±1.78	51.36±2.72	52.39±1.74
FedHERO$_{att}$	34.92±1.00	36.80±1.49	51.71±1.88	54.28±1.62
FedHERO$_{gat}$	34.83±0.81	37.12±1.16	51.45±1.73	53.26±2.43
FedHERO+GLCN	35.11±1.14	37.25±0.85	52.18±3.63	52.54±2.29
FedHERO+GAug	36.65±1.53	37.63±1.00	52.44±2.67	54.57±2.42
FedHERO	**37.83±1.02**	**38.00±0.92**	**52.65±1.33**	**55.63±2.00**

4.5 FedHERO with Various Sharing Mechanisms

We conduct an analysis of various FedHERO variants involving the sharing of different components of the local model, including all model parameters, task model parameters, and none (denoted as FedHERO$_{all}$, FedHERO$_{gcn}$, and FedHERO$_{none}$, respectively). Additionally, we consider whether to incorporate a structure-task model decoupled scheme. For variants without the Dual-Channel (DC), we remove the GNN in the f_{local} channel and feed the learned adjacency matrix and the original matrix into the same GNN.

In Table 5, we observe that the structure-task model decoupling scheme consistently outperforms the model without it, indicating that independently learning the structural knowledge enhances FedHERO's performance across various scenarios. Furthermore, sharing all parameters or the task model results in performance degradation compared to pure local training. This observation underscores that sharing local task model information across heterophily graphs harms FedHERO's performance, since the original graphs have different node neighbor distribution patterns and aggregating of models that capture distinct tendencies would lead to performance degradation for the global model.

4.6 Convergence Analysis

We conduct experiments to compare the convergence speed of FedHERO with other FGL baselines on different datasets. The results, as presented in Figure 4, demonstrate that FedHERO exhibits faster convergence compared to most baselines on both datasets. Notably, FedHERO's convergence speed is comparable to

Table 5: Performance of FedHERO when sharing different modules with server.

Variants	**DC**	**Squirrel**		**Chameleon**	
		METIS M=7	Louvain	METIS M=7	Louvain
FedHERO$_{none}$	-	32.53±0.66	36.85±0.85	48.48±1.12	48.30±1.50
FedHERO$_{all}$	-	29.37±1.18	32.84±0.92	44.09±2.84	44.12±1.98
FedHERO$_{none}$	✓	34.14±1.35	36.25±2.96	50.62±1.54	52.03±1.70
FedHERO$_{all}$	✓	32.04±1.57	34.08±2.14	45.38±1.64	46.39±3.03
FedHERO$_{gcn}$	✓	28.05±0.65	31.13±1.57	42.27±1.67	43.70±1.86
FedHERO	✓	**37.83±1.02**	**38.00±0.92**	**52.65±1.33**	**55.63±2.00**

that of FedStar, since both two methods employ a dual-channel GNN in the client's model. We conclude that FedHERO does not require more steps to converge compared to existing methods, though it introduces additional parameters, which underscores the efficiency of FedHERO.

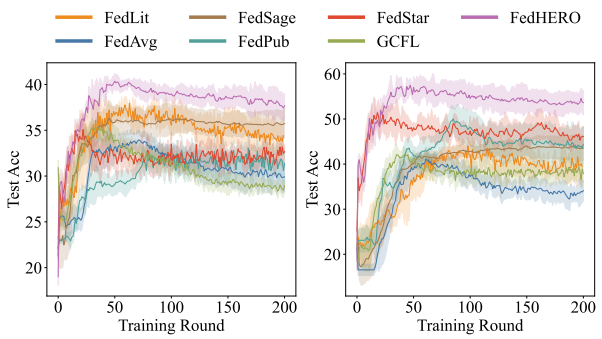


Figure 4: Convergence analysis of FedHERO and baselines on (left) Squirrel and (right) Chameleon datasets.

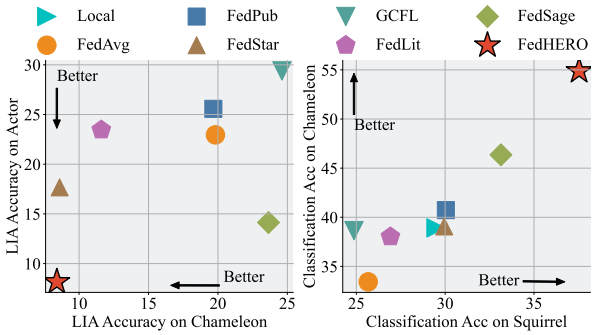


Figure 5: (**Left**) Link inference attack (LIA) accuracies on FedHERO and FGL baseline methods. (**Right**) Performance of FedHERO and FGL baseline methods on noisy graph datasets.

4.7 Privacy Preservation in FedHERO

FedHERO only transmits the components in the f_{global} channel to the server. This subset contains fewer parameters, thus reducing the volume of sensitive information transmitted. Furthermore, we evaluate the leakage of structural information from FedHERO through the link inference attack (LIA) (Gong & Liu, 2018). LIA aims to infer the underlying graph structure from the learned node representations. The LIA attack accuracies on two datasets are presented in Figure 5 (left). In general, LIA struggles to accurately predict the graph structure due to the inherent heterophilic characteristics of the graphs. We can observe that FedHERO offers better privacy protection compared to the FGL baseline approaches by sharing only a portion of the model, and make LIA even harder to predict.

4.8 Robustness Study of FedHERO

We assess the robustness of FedHERO by introducing noise to the graph, flipping each edge with a probability of 0.1. As shown in Figure 5 (right), while baseline FGL methods experience significant performance degradation under such conditions, FedHERO maintains its effectiveness. This resilience is due to FedHERO's unique design: unlike other FGL methods that rely heavily on the original graph structure, FedHERO shares a global structure learner and performs message passing on a generated latent graph. This approach reduces the impact of noise on the original graph's structure, enhancing FedHERO's adaptability to noisy environments. Additionally, FedHERO exhibits reduced reliance on the local graph structure, demonstrating its ability to address the performance deterioration brought by varying neighbor distribution patterns and effectively handle heterophilic graphs.

5 Related Work

Graph Structure Learning. Graph Neural Networks (GNNs) have emerged as a potent tool for analyzing graph-structured data and have found wide applications in various domains (Zhao et al., 2021a; Wu et al., 2019; Kipf & Welling, 2016; Zhu et al., 2019; Liu et al., 2020; Fan et al., 2019; Luan et al., 2022; Yuan et al., 2022; Dwivedi et al., 2023; Shlomi et al., 2020). In order to overcome the limitations of GNNs in effectively propagating features within observed structures, recent efforts have aimed at simultaneous learning of graph structures alongside the GNN model (Jin et al., 2020; Liu et al., 2022b; Li et al., 2024; Liu et al., 2023b; Wu et al., 2022). At the heart of graph structure learning lies an encoding function responsible for modeling optimal graph structures, often represented by edge weights (Luo et al., 2021; Kreuzer et al., 2021; Sun et al.,

2022; Wang et al., 2020; Gasteiger et al., 2019; Yu et al., 2021). These edge weights can be defined through pairwise distances or learnable parameters. For example, GAUGM (Zhao et al., 2021b) calculates edge weights by taking the inner product of node embeddings, introducing no additional parameters. Franceschi et al. (2019) treat each edge as a Bernoulli random variable and optimize graph structures with the GNN. IDGL (Chen et al., 2020), to make full use of information from observed structures for structure learning, employs a multi-head self-attention network to represent pairwise relationships between nodes, while Li et al. (Li et al., 2018a) adopt a metric learning approach based on the Radial Basis Function (RBF) kernel for a similar goal. Moreover, DGM (Kazi et al., 2022) utilizes the Gumbel-Top-k trick to sample edges from a Gaussian distribution and incorporates reinforcement learning, rewarding edges that contribute to correct classifications and penalizing those leading to misclassifications. Additionally, Zhang et al. (2019) present a probabilistic framework that views the input graph as a random sample from a collection modeled by a parametric random graph model. However, while these methods have shown promise, they assume that training and testing nodes originate from the same graph, and they primarily consider a single graph. In contrast, Zhao et al. (2023) address graph structure learning in a cross-graph setting and propose a comprehensive framework for learning a shared structure learner that can generalize to target graphs without necessitating re-training.

Federated Graph Learning. Federated Learning (FL) is a paradigm that facilitates model training across decentralized clients while safeguarding privacy (Li et al., 2020; Liu et al., 2024; Gafni et al., 2022; Wang et al., 2023; Ye et al., 2023; Lei et al., 2023). As the field of FL evolves, a new specialized branch has emerged to address the unique challenges of graph data, known as Federated Graph Learning (FGL) (Liu et al., 2022a). This branch can be categorized into graph-level and node-level methods.

Graph-level FGL methods consider scenarios where each client possesses multiple graph data, as seen in domains like molecular structures. Like traditional FL, it mainly focuses on the issue of non-independent and identically distribution (non-IID) issues across clients (Xie et al., 2021; He et al., 2021; Tan et al., 2023; Zheng et al., 2021; Chen et al., 2021a). On the other hand, node-level FGL assumes that each client holds a single graph, with the objective of addressing node classification tasks. In this variant, each client usually possesses a subgraph of the larger graph, introducing a scenario where there are missing links between subgraphs. To address this issue, Zhang et al. introduced FedSage (Zhang et al., 2021), which involves collaboratively learning a missing neighbor generator across clients. However, FedSage could raise privacy concerns. In contrast, Baek et al. (2023) utilize random graphs as inputs to compute similarities between clients' GNNs, subsequently employing these similarities for weighted averaging on the server side. Additionally, Xie et al. (2023) focus on detecting latent link-types during FGL and differentiating message-passing through various types of links using multiple convolution channels. In this study, we address a specific challenge in node-level FGL, namely the distinction in node neighbor distributions across clients. Our approach involves sharing a structure learning model among clients, enabling the learning of message-passing patterns across the graphs. By leveraging this information, we aim to enhance the predictive performance of the local models.

6 Conclusion

The heterophilic patterns of graphs can result in variations in the distribution of node neighbors among different graphs, leading to performance deterioration for current FGL algorithms. The FedHERO framework effectively addresses the challenge of varying neighbor distribution patterns, resolving it by sharing structure learners across clients. The structure learner would generate latent graphs with similar tendencies for different clients. Clients' GNNs that capture these tendencies could be aggregated and shared among clients to harness FGL's ability. By decoupling the structural learning model from the local classification model, we effectively capture and make available structural insights for global sharing. Additionally, our proposed framework maintains the capacity for personalization based on local graph structure and node features. Experimental results affirm that FedHERO consistently surpasses state-of-the-art methods across various datasets.

References

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In international conference on machine learning, pp. 21–29. PMLR, 2019.

Jinheon Baek, Wonyong Jeong, Jiongdao Jin, Jaehong Yoon, and Sung Ju Hwang. Personalized subgraph federated learning. In International Conference on Machine Learning, pp. 1396–1415. PMLR, 2023.

Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H Chi. Data decisions and theoretical implications when adversarially learning fair representations. arXiv preprint arXiv:1707.00075, 2017.

Smriti Bhagat, Graham Cormode, and S Muthukrishnan. Node classification in social networks. Social network data analytics, pp. 115–148, 2011.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment, 2008(10):P10008, 2008.

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In Proceedings of the AAAI conference on artificial intelligence, volume 35, pp. 3950–3957, 2021.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine, 34(4):18–42, 2017.

Chuan Chen, Weibo Hu, Ziyue Xu, and Zibin Zheng. Fedgl: Federated graph learning framework with global self-supervision. arXiv preprint arXiv:2105.03170, 2021a.

Mingyang Chen, Wen Zhang, Zonggang Yuan, Yantao Jia, and Huajun Chen. Fede: Embedding knowledge graphs in federated setting. In Proceedings of the 10th International Joint Conference on Knowledge Graphs, pp. 80–88, 2021b.

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

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. arXiv preprint arXiv:2006.07988, 2020.

Weilin Cong and Mehrdad Mahdavi. Grapheditor: An efficient graph representation learning and unlearning approach. 2022.

Enyan Dai, Tianxiang Zhao, Huaisheng Zhu, Junjie Xu, Zhimeng Guo, Hui Liu, Jiliang Tang, and Suhang Wang. A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability. arXiv preprint arXiv:2204.08570, 2022.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. Journal of Machine Learning Research, 24(43):1–48, 2023.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In Proceedings of the 3rd innovations in theoretical computer science conference, pp. 214–226, 2012.

Pantelis Elinas, Edwin V Bonilla, and Louis Tiao. Variational inference for graph convolutional networks in the absence of graph data and adversarial settings. Advances in Neural Information Processing Systems, 33:18648–18660, 2020.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In The world wide web conference, pp. 417–426, 2019.

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In International conference on machine learning, pp. 1972–1982. PMLR, 2019.

Tomer Gafni, Nir Shlezinger, Kobi Cohen, Yonina C Eldar, and H Vincent Poor. Federated learning: A signal processing perspective. IEEE Signal Processing Magazine, 39(3):14–41, 2022.

Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. Advances in neural information processing systems, 32, 2019.

Neil Zhenqiang Gong and Bin Liu. Attribute inference attacks in online social networks. ACM Transactions on Privacy and Security (TOPS), 21(1):1–30, 2018.

Chaoyang He, Emir Ceyani, Keshav Balasubramanian, Murali Annavaram, and Salman Avestimehr. Spreadgnn: Serverless multi-task federated learning for graph neural networks. arXiv preprint arXiv:2106.02743, 2021.

Addison Howard, Bernadette Bouchon-Meunier, IEEE CIS, inversion, John Lei, Lynn@Vesta, Marcus2010, and Hussein Abbass. Ieee-cis fraud detection, 2019. URL `https://kaggle.com/competitions/ieee-fraud-detection`.

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

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 66–74, 2020.

George Karypis and Vipin Kumar. Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. 1997.

Anees Kazi, Luca Cosmo, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael M Bronstein. Differentiable graph module (dgm) for graph convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(2):1606–1617, 2022.

Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. arXiv preprint arXiv:2204.04879, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. Advances in Neural Information Processing Systems, 34: 21618–21629, 2021.

Runze Lei, Pinghui Wang, Junzhou Zhao, Lin Lan, Jing Tao, Chao Deng, Junlan Feng, Xidian Wang, and Xiaohong Guan. Federated learning over coupled graphs. IEEE Transactions on Parallel and Distributed Systems, 34(4):1159–1172, 2023.

Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In Proceedings of the AAAI conference on artificial intelligence, volume 32, 2018a.

Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In Proceedings of the AAAI conference on artificial intelligence, volume 32, 2018b.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. IEEE signal processing magazine, 37(3):50–60, 2020.

Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In International Conference on Machine Learning, pp. 13242–13256. PMLR, 2022.

Zhixun Li, Xin Sun, Yifan Luo, Yanqiao Zhu, Dingshuo Chen, Yingtao Luo, Xiangxin Zhou, Qiang Liu, Shu Wu, Liang Wang, et al. Gslb: the graph structure learning benchmark. Advances in Neural Information Processing Systems, 36, 2024.

Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 338–348, 2020.

Rui Liu, Pengwei Xing, Zichao Deng, Anran Li, Cuntai Guan, and Han Yu. Federated graph neural networks: Overview, techniques and challenges. arXiv preprint arXiv:2202.07256, 2022a.

Rui Liu, Pengwei Xing, Zichao Deng, Anran Li, Cuntai Guan, and Han Yu. Federated graph neural networks: Overview, techniques, and challenges. IEEE Transactions on Neural Networks and Learning Systems, 2024.

Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. Towards unsupervised deep graph structure learning. In Proceedings of the ACM Web Conference 2022, pp. 1392–1403, 2022b.

Yixin Liu, Yizhen Zheng, Daokun Zhang, Vincent CS Lee, and Shirui Pan. Beyond smoothing: Unsupervised graph representation learning with edge heterophily discriminating. In Proceedings of the AAAI conference on artificial intelligence, volume 37, pp. 4516–4524, 2023a.

Zhaowei Liu, Dong Yang, Yingjie Wang, Mingjie Lu, and Ranran Li. Egnn: Graph structure learning based on evolutionary computation helps more in graph neural networks. Applied Soft Computing, 135:110040, 2023b.

Guannan Lou, Yuze Liu, Tiehua Zhang, and Xi Zheng. Stfl: A temporal-spatial federated learning framework for graph neural networks. arXiv preprint arXiv:2111.06750, 2021.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. Advances in neural information processing systems, 35:1362–1375, 2022.

Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. Learning to drop: Robust graph neural network via topological denoising. In Proceedings of the 14th ACM international conference on web search and data mining, pp. 779–787, 2021.

Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. Learning disentangled representations for recommendation. Advances in neural information processing systems, 32, 2019.

Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? arXiv preprint arXiv:2106.06134, 2021.

Haitao Mao, Zhikai Chen, Wei Jin, Haoyu Han, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. Demystifying structural disparity in graph neural networks: Can one size fit all? arXiv preprint arXiv:2306.01323, 2023.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics, pp. 1273–1282. PMLR, 2017.

Guangxu Mei, Ziyu Guo, Shijun Liu, and Li Pan. Sgnn: A graph neural network based federated learning approach by hiding structure. In 2019 IEEE International Conference on Big Data (Big Data), pp. 2560–2568. IEEE, 2019.

Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. Query-driven active surveying for collective classification. In 10th international workshop on mining and learning with graphs, volume 8, pp. 1, 2012.

Hieu V Nguyen and Li Bai. Cosine similarity metric learning for face verification. In Asian conference on computer vision, pp. 709–720. Springer, 2010.

Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In Proceedings of the 16th international conference on World Wide Web, pp. 201–210, 2007.

Hao Peng, Haoran Li, Yangqiu Song, Vincent Zheng, and Jianxin Li. Differentially private federated knowledge graphs embedding. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 1416–1425, 2021.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. Journal of Complex Networks, 9(2):cnab014, 2021.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. IEEE transactions on neural networks, 20(1):61–80, 2008.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. AI magazine, 29(3):93–93, 2008.

Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. Machine Learning: Science and Technology, 2(2):021001, 2020.

Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Xingcheng Fu, Cheng Ji, and S Yu Philip. Graph structure learning with variational information bottleneck. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, pp. 4165–4174, 2022.

Susheel Suresh, Vinith Budde, Jennifer Neville, Pan Li, and Jianzhu Ma. Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 1541–1551, 2021.

Yue Tan, Yixin Liu, Guodong Long, Jing Jiang, Qinghua Lu, and Chengqi Zhang. Federated learning on non-iid graphs via structural knowledge sharing. In Proceedings of the AAAI conference on artificial intelligence, volume 37, pp. 9953–9961, 2023.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 807–816, 2009a.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 807–816, 2009b.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In International Conference on Learning Representations, 2018.

Bin Wang, Jun Fang, Hongbin Li, Xiaojun Yuan, and Qing Ling. Confederated learning: Federated learning with decentralized edge servers. IEEE Transactions on Signal Processing, 71:248–263, 2023.

Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. Am-gcn: Adaptive multi-channel graph convolutional networks. In Proceedings of the 26th ACM SIGKDD International conference on knowledge discovery & data mining, pp. 1243–1253, 2020.

Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. Advances in Neural Information Processing Systems, 35: 27387–27401, 2022.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In Proceedings of the AAAI conference on artificial intelligence, volume 33, pp. 346–353, 2019.

Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. Graph information bottleneck. Advances in Neural Information Processing Systems, 33:20437–20448, 2020a.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems, 32(1):4–24, 2020b.

Feng Xia, Ke Sun, Shuo Yu, Abdul Aziz, Liangtian Wan, Shirui Pan, and Huan Liu. Graph learning: A survey. IEEE Transactions on Artificial Intelligence, 2(2):109–127, 2021.

Shunxin Xiao, Shiping Wang, Yuanfei Dai, and Wenzhong Guo. Graph neural networks in node classification: survey and evaluation. Machine Vision and Applications, 33(1):4, 2022.

Han Xie, Jing Ma, Li Xiong, and Carl Yang. Federated graph classification over non-iid graphs. Advances in Neural Information Processing Systems, 34:18839–18852, 2021.

Han Xie, Li Xiong, and Carl Yang. Federated node classification over graphs with latent link-type heterogeneity. In Proceedings of the ACM Web Conference 2023, pp. 556–566, 2023.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826, 2018a.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In International conference on machine learning, pp. 5453–5462. PMLR, 2018b.

Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In 2022 IEEE International Conference on Data Mining (ICDM), pp. 1287–1292. IEEE, 2022.

Liang Yang, Mengzhe Li, Liyang Liu, Chuan Wang, Xiaochun Cao, Yuanfang Guo, et al. Diverse message passing for attribute with heterophily. Advances in Neural Information Processing Systems, 34:4751–4763, 2021.

Tianmeng Yang, Yujing Wang, Zhihan Yue, Yaming Yang, Yunhai Tong, and Jing Bai. Graph pointer neural networks. In Proceedings of the AAAI conference on artificial intelligence, volume 36, pp. 8832–8839, 2022.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In International conference on machine learning, pp. 40–48. PMLR, 2016.

Rui Ye, Zhenyang Ni, Chenxin Xu, Jianyu Wang, Siheng Chen, and Yonina C Eldar. Fedfm: Anchor-based feature matching for data heterogeneity in federated learning. IEEE Transactions on Signal Processing, 2023.

I-Cheng Yeh and Che-hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert systems with applications, 36(2):2473–2480, 2009.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 974–983, 2018.

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

Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. IEEE transactions on pattern analysis and machine intelligence, 45(5):5782–5799, 2022.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. arXiv preprint arXiv:1907.04931, 2019.

Kai Zhang, Yu Wang, Hongyi Wang, Lifu Huang, Carl Yang, Xun Chen, and Lichao Sun. Efficient federated learning on knowledge graphs via privacy-preserving relation embedding aggregation. arXiv preprint arXiv:2203.09553, 2022.

Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. Subgraph federated learning with missing neighbor generation. Advances in Neural Information Processing Systems, 34:6671–6682, 2021.

Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Ustebay. Bayesian graph convolutional neural networks for semi-supervised classification. In Proceedings of the AAAI conference on artificial intelligence, volume 33, pp. 5829–5836, 2019.

Jianan Zhao, Xiao Wang, Chuan Shi, Binbin Hu, Guojie Song, and Yanfang Ye. Heterogeneous graph structure learning for graph neural networks. In Proceedings of the AAAI conference on artificial intelligence, volume 35, pp. 4697–4705, 2021a.

Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. Data augmentation for graph neural networks. In Proceedings of the aaai conference on artificial intelligence, volume 35, pp. 11015–11023, 2021b.

Wentao Zhao, Qitian Wu, Chenxiao Yang, and Junchi Yan. Graphglow: Universal and generalizable structure learning for graph neural networks. arXiv preprint arXiv:2306.11264, 2023.

Longfei Zheng, Jun Zhou, Chaochao Chen, Bingzhe Wu, Li Wang, and Benyu Zhang. Asfgnn: Automated separated-federated graph neural network. Peer-to-Peer Networking and Applications, 14(3):1692–1704, 2021.

Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. Graph neural networks for graphs with heterophily: A survey. arXiv preprint arXiv:2202.07082, 2022.

Yizhen Zheng, He Zhang, Vincent Lee, Yu Zheng, Xiao Wang, and Shirui Pan. Finding the missing-half: Graph complementary learning for homophily-prone and heterophily-prone graphs. In International Conference on Machine Learning, pp. 42492–42505. PMLR, 2023.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. AI open, 1:57–81, 2020.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. Advances in neural information processing systems, 33:7793–7804, 2020.

Mengxiao Zhu and Haogang Zhu. Mixedad: A scalable algorithm for detecting mixed anomalies in attributed graphs. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pp. 1274–1281, 2020.

Rong Zhu, Kun Zhao, Hongxia Yang, Wei Lin, Chang Zhou, Baole Ai, Yong Li, and Jingren Zhou. Aligraph: A comprehensive graph neural network platform. arXiv preprint arXiv:1902.08730, 2019.

Wei Zhu, Jiebo Luo, and Andrew D White. Federated learning of molecular properties with graph neural networks in a heterogeneous setting. Patterns, 3(6), 2022.

A Experiment Setups

In our experiments, we utilize seven distinct benchmark datasets, the specifics of which are outlined in Table 6. This table provides detailed statistics for each dataset, including the number of nodes, edges, classes, and node feature dimensions. Additionally, we describe the process of partitioning the original graphs into multiple subgraphs to create semi-synthetic federated graph learning datasets.

For graph partitioning, we employ well-established algorithms commonly used in federated graph learning. Specifically, we utilize the METIS algorithm (Karypis & Kumar, 1997) and the Louvain algorithm (Blondel et al., 2008). With METIS, we ensure an even distribution of nodes across subsets, with each subset representing a client's data in a federated learning context. Using Louvain, we follow the methodology of Baek et al. (2023) and (Zhang et al., 2021). We initially partition the graph and select subgraphs containing a minimum of 50 nodes. The smaller subgraphs are then randomly merged into these larger ones to guarantee adequate training data. For example, if Louvain results in 20 subgraphs but only 5 have over 50 nodes, we consolidate the remaining 15 subgraphs into these 5, effectively simulating a federated learning scenario with 5 clients.

We further provide the description of real-world datasets. The syn-cora dataset (Zhu et al., 2020) generates nodes and edges based on the expected homophily ratio within the graph, with feature vectors of nodes in each class derived by sampling from the corresponding class in the Cora dataset (Yang et al., 2016). The ieee-fraud dataset comprises transaction records for products, with edges determined by the proximity of transaction occurrences. The objective is to predict fraudulent transactions (Howard et al., 2019). Meanwhile, the credit dataset has 30,000 nodes, each representing an individual, and their connections are determined by the similarity of their spending and payment patterns (Yeh & Lien, 2009). The objective is to predict whether an individual will default on their credit card payment or not.

Table 6: Dataset statistics

Dataset	Squirrel	Chameleon	Actor	Flickr	syn-cora	ieee-fraud	credit
Nodes	5,201	2,277	7,600	89,250	14,900	144,233	30,000
Edges	198,493	31,421	15,009	899,756	29,674	28,053,717	1,436,858
Features	2,089	2,235	932	500	1433	371	13
Classes	5	5	5	7	5	2	2

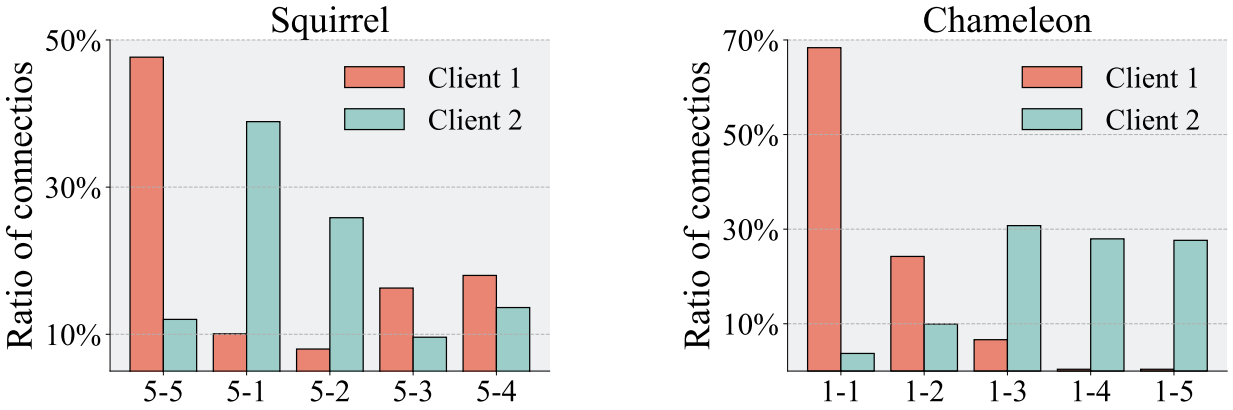
We follow the previous work (Mao et al., 2023) to compute the homophily ratios of both the original graphs and the averaged homophily ratios of partitioned graphs. As in Table 7, the partitioned graphs exhibit comparable homophily ratios when compared to the original graphs. It means that the partitioning process does not compromise the heterophilic property of the graphs.

Table 7: Homophily ratio of different partitioning methods.

Dataset	Squirrel	Chameleon	Actor	Flickr
METIS	0.22±0.02	0.26±0.05	0.17±0.02	0.33±0.04
Louvain	0.21±0.02	0.27±0.06	0.19±0.04	0.32±0.04
Origin	0.22	0.25	0.22	0.32

B Neighbor node distributions of clients in two datasets

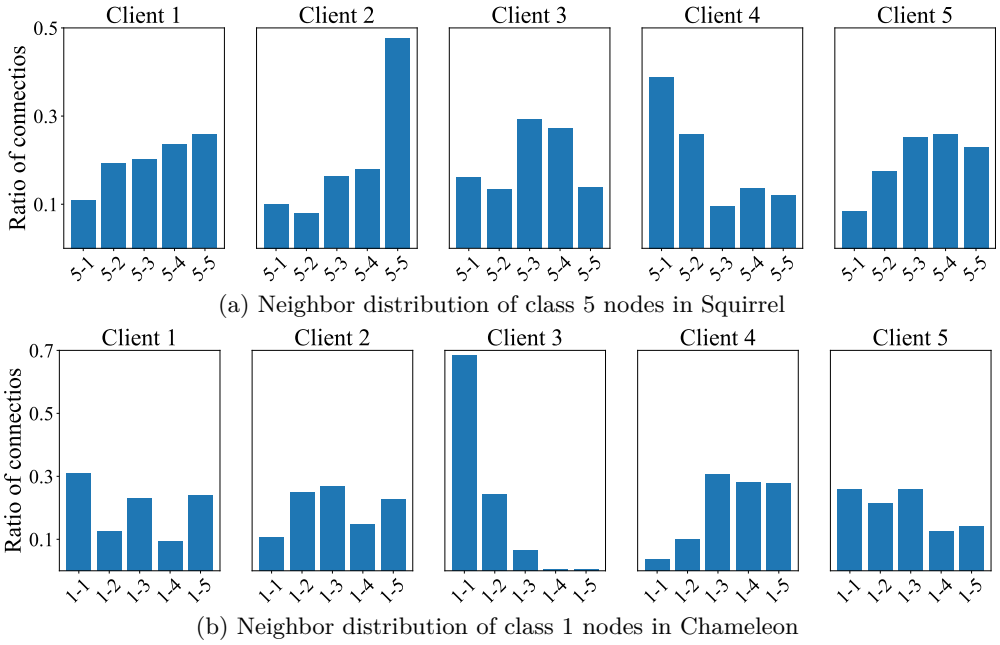
In this section, we also give an example in two real-world datasets: Squirrel and Chameleon (Rozemberczki et al., 2021). In these datasets, nodes correspond to web pages, and edges represent hyperlinks between them. To simulate the FL scenario, we apply the community detection algorithm METIS (Karypis & Kumar, 1997) to partition these graphs, and variations in the structures of the resulting subgraphs also arise. As illustrated in Figure 6, we compare the neighbor distributions of two clients in two datasets, respectively. (The neighbor distributions across all clients can refer to Figure 7.) Each bar illustrates, for a specific class of nodes, the



(a) connections of class 5 nodes (b) connections of class 1 nodes

Figure 6: Neighbor node distributions of two clients in two datasets respectively. Each bar represents the ratio of edges connecting nodes of one class to the total number of edges with nodes of that class as an endpoint. For example, in Figure (a), the orange bar at '5-1' indicates the proportion of edges from class 5 to class 1 nodes relative to total edges originating from class 5 nodes in client 1.

ratio of edges connecting them to other types of nodes of one type to the total number of edges serving this class node as an endpoint. For instance, in Figure 6 (a), the orange bar with the x-axis '5-1' signifies the proportion of edges linking class 5 nodes to class 1 nodes in relation to the overall number of edges originating from class 5 nodes. The results indicate distinct neighbor distributions among different clients. For instance, in the Chameleon dataset, almost 70% of edges originating from class 1 nodes are intra-class connections in client 1, implying class 1 nodes tend to establish hyperlinks with similar web pages. Conversely, less than 10% of the neighbors of class 1 nodes are nodes in the same class in client 2, indicating a preference for establishing hyperlinks with diverse web pages. To safeguard data privacy, the structural details of subgraphs cannot be directly shared between clients. Consequently, this leads to distinct message-passing patterns of information transfer between these subgraphs. Traditional FL methods like FedAvg (McMahan et al., 2017) simply aggregate GNN models, potentially hindering the global model's ability to account for the aggregation discrepancies among various subgraphs effectively.



(a) Neighbor distribution of class 5 nodes in Squirrel

(b) Neighbor distribution of class 1 nodes in Chameleon

Figure 7: Neighbor node distributions of clients in two datasets split by METIS.

C Case study

In this section, we present a series of case studies designed to further illustrate the practical applications and effectiveness of FedHERO under diverse conditions. These case studies encompass a range of scenarios, each highlighting different aspects of our approach.

C.1 Overlapping subgraphs scenario

Despite the commonality of the non-overlapping node assumption, some research investigates scenarios with overlapping nodes. We have followed the data partitioning approach from (Baek et al., 2023) in our experiments, focusing on the Flickr and Actor datasets. The results, as illustrated in Table 8, show FedHERO consistently outperforming most baselines across these datasets. This indicates its robust capability to manage overlapping scenarios effectively. We theorize that overlapping subgraphs provide client subgraphs with additional connections, hinting at the potential for developing FL algorithms specifically designed to cater to overlapping environments.

Table 8: Results on the overlapping subgraphs scenario.

Method	Local	FedAvg	FedPub	FedStar	GCFL	FedLit	FedSage	FedHERO
Flickr	55.70±0.25	55.34±0.69	55.50±0.31	55.92±0.42	56.47±0.21	56.64±0.50	56.70±0.42	**56.74±0.30**
Actor	32.46±0.38	32.30±1.21	28.67±0.89	27.98±0.77	42.48±1.23	42.50±0.66	41.57±0.89	**43.46±1.11**

C.2 Latent graph generation

In graph structure learning, there are various methods for generating latent graphs. In our approach, we define $\tilde{a}_{(i,j)}$ as the learned edge weight between nodes u and v, and subsequently apply a post-processing step to form a kNN graph (where each node is connected to up to k neighbors) as the latent graph. This choice is informed by the differentiability of the top k function, which aids in calculating parameter gradients and updating the model in FedHERO. We also examine the influence of another prevalent latent graph generation method (Wu et al., 2020a; Elinas et al., 2020; Zhao et al., 2023). This method treats each edge in the latent graph as a Bernoulli random variable, resulting in edge distributions represented by a product of $N \times N$ independent Bernoulli variables.

Table 9 presents an analysis of FedHERO's performance when employing the Bernoulli sampling method for latent graph generation during inference. Using $\tilde{a}_{(i,j)}$ as a Bernoulli parameter introduces greater variance, potentially leading to less stable model performance. Despite this, FedHERO still surpasses most baselines, as evidenced by the comparative results in Table 3. Our experiments also suggest that an asymmetric latent graph may impact model performance. Investigating optimal strategies for utilizing the Bernoulli sampling method in latent graph generation offers a promising direction for future research and development in our work.

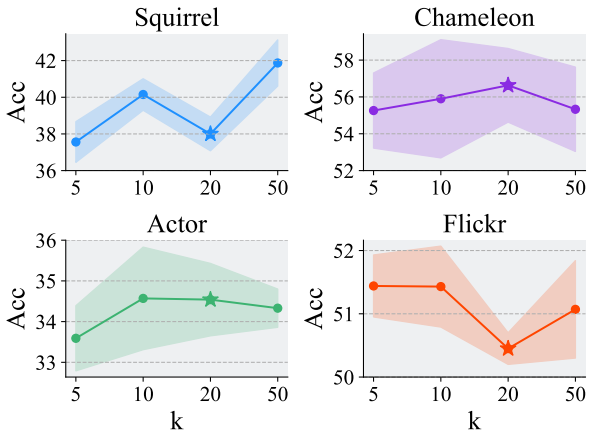


Figure 8: The sensitivity study of hyperparameter k. The star point in the figure corresponds to the default result reported in the main text.

Table 9: Results on the different latent graph generation.

Datasets	Squirrel		Chameleon	
	METIS M=7	Louvain	METIS M=7	Louvain
Bernoulli	33.62±1.81	36.97±4.43	48.82±3.52	51.59±2.99
top k	**36.08±0.87**	**38.00±0.92**	**52.65±1.33**	**55.63±2.00**

To assess the impact of the node degree k in the latent graph, we conducted tests with k set to 5, 10, 20, and 50. Generally, a larger k value results in a denser latent graph. The outcomes of these tests are illustrated in Figure 8. When these findings are combined with the baseline methods' performance from Table 2 in the main paper, we can deduce the following:

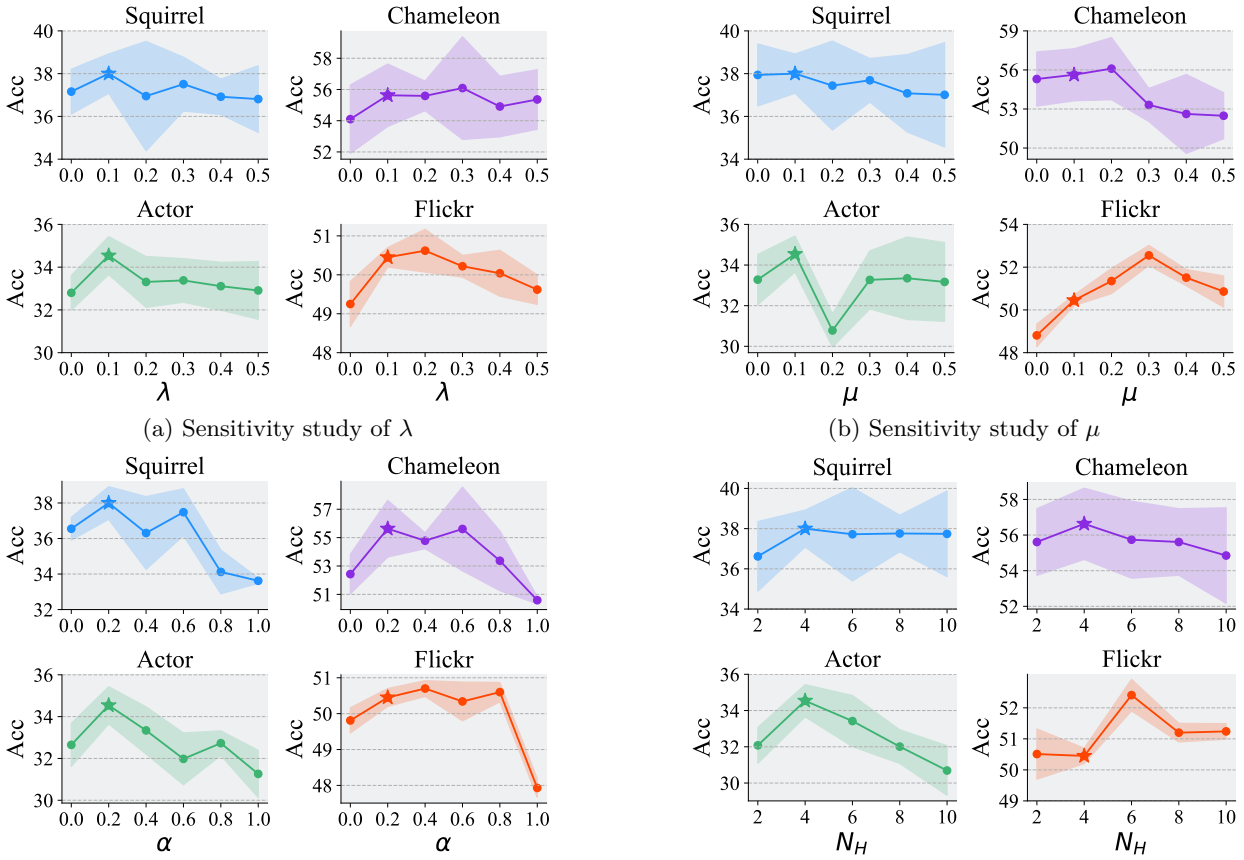
- Across varying values of k, FedHERO consistently outperforms all the baseline methods.
- For datasets with a smaller average node degree, such as Actor and Flickr, selecting a relatively smaller k value is advantageous. Conversely, for datasets with a larger average node degree, like Squirrel and Chameleon, opting for a larger k value is preferable.

These insights provide valuable guidance for choosing appropriate k values based on the characteristics of the dataset, thereby optimizing the performance of FedHERO.

C.3 Federated graph learning baselines

We provide detailed explanations of the federated graph learning baselines employed in our experiments. These include:

- GCFL (Xie et al., 2021): It incorporates a gradient sequence-based clustering mechanism using dynamic time warping. This mechanism enables multiple data owners with non-IID structured graphs and diverse features to collaboratively train robust graph neural networks.
- FedStar (Tan et al., 2023): It relies on structural embeddings and a feature-structure decoupled graph neural network to address non-IID data challenges through shared structural knowledge. Within the FL framework, it exclusively exchanges the structure encoder. This allows clients to collectively learn global structural knowledge while maintaining personalized learning of node representations based on their features.
- FedLit (Xie et al., 2023): It tackles the variability in link characteristics within graphs, where ostensibly uniform links may convey varying degrees of similarity or significance. It dynamically identifies latent link types through an EM-based clustering algorithm during federated learning and adapts message-passing strategies using multiple convolution channels tailored to different link types.
- FedPub (Baek et al., 2023): It leverages functional embeddings of the local GNNs with random graphs as inputs to calculate similarities between them. These similarities are then employed for weighted averaging during server-side aggregation. Additionally, the method learns a personalized sparse mask at each client, allowing for the selection and update of only the subset of aggregated parameters relevant to the specific subgraph.
- FedSage (Zhang et al., 2021): It introduces NeighGen, an innovative missing neighbor generator integrated with federated training procedures. This endeavor is geared towards achieving a universal node classification model within a distributed subgraph framework, eliminating the need for direct data sharing.



(a) Sensitivity study of λ

(b) Sensitivity study of μ

(c) Sensitivity study of α

(d) Sensitivity study of N_H

Figure 9: Performance of FEDHERO with different hyperparameter settings. The star point in the figure corresponds to the default result reported in the main text.

D Influence of Hyperparameters in FedHERO

D.1 Hyperparameter Settings.

The architecture of the GNN models consists of a feature projection layer, two graph convolutional layers, and a classifier layer. We employ Adam optimizer (Kingma & Ba, 2014) with a learning rate of 0.005. We apply a consistent hyperparameter setting in our main results. Specifically, we set $\alpha = 0.2$, $\mu = \lambda = 0.1$, $k = 20$, and $N_H = 4$. The training process involves 200 communication rounds, with each local training epoch lasting 1 iteration. The nodes are randomly distributed into five groups. For each experiment, one group is chosen as the test set, while the remaining groups are utilized for training and validation. We perform five experiments on each dataset and report the models' average performance. For the ieee-fraud and credit datasets, we present the mean test Area Under the Curve (AUC), accounting for the imbalanced labels. For the remaining datasets, we use mean test accuracy across clients for evaluation.

In an ideal scenario, a grid search for hyperparameter optimization would be conducted to determine the optimal combination for each set of experiments. It's noteworthy that our experimental results consistently demonstrate the superior performance of FEDHERO compared to all baseline methods, even without hyperparameter tuning. Additionally, we conducted an investigation into the impact of these hyperparameters under the Louvain data partitioning method to provide recommended values for these aggregation and smoothing hyperparameters and their potential impact. The results are illustrated in Figure 9.

D.2 Influence of Smoothing Hyperparameters λ and μ

No distinct trend related to the intrinsic nature of the data was observed for λ and μ. However, it is noteworthy that removing regularization on structures (i.e., setting λ or μ to 0) resulted in more pronounced performance degradation. In fact, the regularization loss plays a constructive role in offering guidance for structure learning, as highlighted in prior work (Zhao et al., 2023). Conversely, increasing λ or μ to a certain value proved beneficial in enhancing the performance of FedHERO. However, further increases had a negative impact on the model's focus on the classification task, leading to a decline in performance.

D.3 Influence of Aggregation Hyperparameters α

For the Squirrel, Chameleon, and Actor datasets, optimal results were observed at $\alpha = 0.2$, while $\alpha = 0.4$ exhibited the best performance for the Flickr dataset, as illustrated in Figure 9(c). This variation in optimal α values is attributed to the homophily ratios of the datasets. According to Mao et al. (2023) and Kim & Oh (2022), the homophily ratios for Squirrel, Chameleon, and Actor are 0.22, 0.25, and 0.22, respectively. In contrast, Flickr has a higher homophily ratio of 0.32. Given Flickr's relatively elevated homophily ratio, employing a larger α to increase the proportion of local models proves beneficial, effectively leveraging the structural information within the subgraph. Conversely, for datasets with lower homophily ratios, superior performance is achieved by placing more emphasis on the homophily properties introduced by the global structure learner. In summary, we posit that, for graph data with a larger homophily ratio (indicating lower heterophily), higher values of α can be advantageous.

D.4 Influence of the Number of Heads N_H

Figure 9(d), we investigate FedHERO's performance under varying values of N_H. We observe that increasing N_H up to a certain point enhances FedHERO's performance. This improvement can be attributed to the aggregation of multi-head results, bolstering the model's ability to capture underlying influences between nodes from diverse perspectives.

However, beyond a certain threshold, further increases in N_H do not yield additional performance improvements. This phenomenon could be attributed to factors such as the saturation of connections between nodes, potential overfitting concerns, and the increased time and computing resources required with larger N_H.

E Comparative Analysis with FedStar

This subsection offers a detailed comparison between FedHERO and FedStar due to their similarity. We observe that the design of FedHERO shares similarities with the concept in FedStar (Tan et al., 2023), which involves decoupling the GNN into feature and structure encoders. However, FedHERO distinguishes itself from FedStar in both purpose and the utilization of structural information. In FedStar, structural information is integrated into features to capture additional domain-invariant characteristics (Tan et al., 2023). Conversely, FedHERO is tailored for scenarios where graphs in clients originate from the same domain but possess distinct node neighbor distributions. Its objective is to learn generally applicable message-passing patterns that are conducive to different heterophilic graphs and can be shared across clients. While FedStar introduces degree-based and random walk-based structure embedding techniques to extract structural information for each node, these embeddings might prove simplistic for the problem outlined in the Introduction. For instance, if one node has three neighbor nodes with the same class and another node has three neighbor nodes with different classes, the degree-based method would yield identical embeddings for both nodes. Additionally, in situations where two graphs share the same adjacency matrix but feature different node class distributions, the random walk-based method might not reach its potential. In contrast, the structure learning model in FedHERO is designed to generate a desired graph structure, bypassing the challenges encountered by the structure embedding method in FedStar. This is corroborated by the experimental results outlined in the Experiment Section.