Model-Heterogeneous Federated Graph Learning with Prototype Propagation Supplementary Material

Author Name
Affiliation
email@example.com

A Pseudo Algorithm of FedPPN

Algorithm 1 FedPPN

Input: Total number of clients M, number of FL rounds T, number of local update epochs Q, number of propagation layers K, propagation weight λ, learning rate η.

Server Executes:

```
Initialize global prototype set P̄ for all classes.
for each round t = 0 to T − 1 do
    for each client i in parallel do
        {P_i^s}_{s=1}^C ← LocalUpdate(i, P̄);
    end for
    Update global prototype by Eq. (4).
end for
```

LocalUpdate$(i, \bar{\mathbf{P}})$:

```
for each local epoch e = 0, 1, ···, Q do
    Compute local GNN's prediction by Eq. (10).
    Compute prototype-based prediction by Eq. (9).
    Compute ensemble prediction by Eq. (11).
    Update local model according to Eq. (14).
end for
Compute local prototype set {P_i^s}_{s=1}^C by Eq. (3).
return {P_i^s}_{s=1}^C
```

B Detailed Experimental Setups

This section presents more details of our experiments, including dataset statistics in Subsection B.1, the essential ideas and hyper-parameters search for baselines in Subsection B.2, detailed model architecture setups in Subsection B.3, and the platform for our experiment environment in Subsection B.4.

B.1 Datasets

Following [Huang *et al.*, 2023], we conduct experiments on three widely used benchmark datasets: Cora [Sen *et al.*, 2008], Citeseer [Sen *et al.*, 2008], and PubMed [Namata *et al.*, 2012]. For all datasets, nodes are randomly split into train/valid/test nodes with a ratio of 60%:20%:20%.

To simulate FGL scenarios, we adopt the Louvain [Blondel *et al.*, 2008] algorithm (used in [Zhang *et al.*, 2021; Huang *et al.*, 2023; Yao *et al.*, 2023]) to partition the original graph into multiple subgraphs, the number of which is the same as the number of FL clients. Then, we allocate a different subgraph to each client as their local graph data. The subgraph partition by the Louvalin is non-overlapping, i.e., there is no intersection between the nodes of each client. Besides, after partition, all edges between subgraphs will lost. Here, we called such missing cross-subgraph (cross-client) edges as missing edegs.

Table 1 reports detailed statistics of each dataset's original graph and subgraphs. Specifically, the table's upper section presents the statistics of the original graph, including the number of nodes, edges, features, and classes. In comparison, the lower section of the table shows the average number of nodes and edges for each subgraph. Furthermore, we also report the number of missing edges and missing rates under the setting of different numbers of clients.

B.2 Baselines and Hyper-parameter

Here, we present details about our chosen baselines and their key hyper-parameters that we searched for a fair comparison.

1. **Local.** Each client trains its GNN model using the local private dataset without a federated learning process.
2. **FML** (ArXiv, [Shen *et al.*, 2020]). It is a pioneering model-heterogeneous FL method based on Deep Mutual Learning (DML) technology [Zhang *et al.*, 2018]. In FML, each client trains two models: a customized local model and a shared global model. The local model is only trained locally, while the global model will be aggregated on the server at each round using FedAvg [McMahan *et al.*, 2017]. Each client will train the two models simultaneously based on a mutual distillation loss function. In our experiment, we choose a two-layer GCN [Kipf and Welling, 2017] with 64 hidden dimensions as the global model. To obtain the optimal accuracy, we perform a grid search on the two key hyper-parameters α and β, from $[0.1, 0.3, 0.5, 0.7, 0.9]$.
3. **FedKD** (Nature Communication, [Wu *et al.*, 2022]). This method is also a DML-based approach. Specifically, FedKD is improved from the FML. It uses an adaptive distillation weight coefficient to replace the weight coefficient of FML. Besides, it additionally adds an adaptive hidden loss to improve performance further. In our experiment, we use a two-layer GCN, the same as the FML, as the shared global model. We specify

Original Dataset	Cora			CiteSeer			PubMed		
# Nodes	2708			3327			19717		
# Edges	10556			9104			88648		
# Features	1433			3703			500		
# Classes	7			6			3		
After Splitting	**3 clients**	**5 clients**	**10 clients**	**3 clients**	**5 clients**	**10 clients**	**3 clients**	**5 clients**	**10 clients**
Avg($\|\mathcal{V}_i\|$)	921	561	290	1129	685	352	6592	3960	1957
Avg($\|\mathcal{E}_i\|$)	3198	1895	902	2875	1739	863	26592	15476	7046
# Missing Edges	962	1080	1532	478	406	468	8870	11268	18184
# Missing Rate	9.1%	10.2%	14.5%	5.0%	4.5%	5.1%	10.0%	12.7%	20.5%

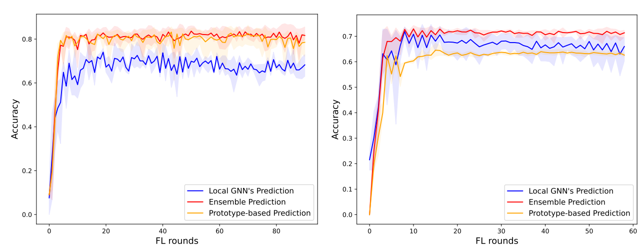
Table 1: **Dataset statistics**. Avg($|\mathcal{V}_i|$) represents the average number of nodes for clients, and Avg($|\mathcal{E}_i|$) represents the average number of edges for clients. Please see Sec B.1 for more details.

the hyper-parameters $T_{start} = 0.95$ and $T_{end} = 0.98$, consistent with the default setup in the official code implementation of FedKD.

4. **FedProto** (AAAI, [Tan *et al.*, 2022a]). This method is the most popular and representative prototype-based approach. It proposes to transmit only class prototypes between clients and the server, significantly reducing communication overhead. To improve model performance, FedProto proposes to align clients' local prototypes by adding a regulation term for each client. We perform a grid search on the hyper-parameter λ from [0.1, 1, 10], where λ is the weight coefficient of the regulation term in each client's local loss function.

5. **FedPCL** (NIPS, [Tan *et al.*, 2022b]). It is a follow-up method from the same first author of FedProto. Compared to FedProto, clients in FedPCL will additionally share their local prototype set with all other clients. In other words, each client will receive a global prototype set and all other clients' local prototypes from the server. Besides, FedPCL designed two novel prototype-based contrastive loss terms to replace the original regulation term in FedProto. In FedPCL, each client's local model includes one (or multiple) pre-trained backbone network(s) as a feature extractor. To ensure a fair comparison, we remove the pre-trained backbone network in our implementation and keep each client's local model of FedPCL consistent with other baselines.

6. **FedGH** (ACM MM, [Yi *et al.*, 2023]). This method is an emerging prototype-based approach. It proposes to train a shared global classification header (a.k.a. projection header) on the server. Specifically, the global classification header is trained based on the clients' local prototypes and corresponding class labels. In each FL round, clients will upload their local prototypes to the server and receive trained classification from the server. Then, each client replaces its local classification header with the received global classification header. In our implementation, we fixed the server-side optimizer to be consistent with each client's local optimizer (the same as the original setup of FedGH).



(a) Cora (b) Citeseer

Figure 1: **Node classification accuracy** for local GNN's, prototype-based, and ensemble prediction. Please see Sec C.3 for more details.

B.3 Model Architectures

Here, we introduce the specific architectural configurations of the models utilized in our experiment. Our primary experiment involves three widely recognized Graph Neural Network (GNN) models: Graph Convolutional Network (GCN) [Kipf and Welling, 2017], Graph Attention Network (GAT) [Velickovic *et al.*, 2018], and Generalized PageRank GNN (GPR-GNN) [Chien *et al.*, 2021]. Specifically, we assign a unique local GNN encoder from these three models to each client and vary the number of layers in each client's GNN encoder to improve model heterogeneity. Additionally, we specify all clients' prediction headers as Fully Connected (FC) layers, aligning with the prior FL research practices [Yi *et al.*, 2023; Tan *et al.*, 2022a]. To sum up, Table 2 outlines the architectural specifications implemented in our main experiment as well as in the additional supplemental experiments (Sec C.1). Furthermore, Table 3 details the specific GNN encoders assigned to each client in our main experiment under the setups of a total number of clients 3, 5, and 10.

B.4 Platform

All models are trained on a Nvidia GeForce RTX 3090 24GB GPU in Ubuntu 20.04.1x64 with a 16-core Intel CPU and 187GB RAM. Python 3.9 and Pytorch 1.10.1 are adopted in the experiments. We utilize the FederatedScope-GNN [Wang *et al.*, 2022] framework to imitate the FGL setting.

Model	Forward pipeline for GNN Encoder	Prediction Header
1-layer GAT	GATConv(d, 64)	FC(64, C)
2-layer GAT	GATConv(d, 64), Drop, ReLU, GATConv(64, 64)	FC(64, C)
2-layer GAT_2	GATConv(d, 256), Drop, ReLU, GATConv(64, 64)	FC(64, C)
3-layer GAT	GATConv(d, 64), Drop, ReLU, GATConv(64, 64), Drop, ReLU, GATConv(64, 64)	FC(64, C)
1-layer GCN	GCNConv(d, 64)	FC(64, C)
2-layer GCN	GCNConv(d, 64) ,Drop, ReLU, GCNConv(64, 64)	FC(64, C)
2-layer GCN_2	GCNConv(d, 128) ,Drop, ReLU, GCNConv(128, 64)	FC(64, C)
3-layer GCN	GCNConv(d, 64), Drop, ReLU, GCNConv(64, 64), Drop, ReLU, GCNConv(64, 64)	FC(64, C)
8-layer GPR-GNN	Drop, FC(d, 64), ReLU, Drop, FC(64, 64), 8-layer GPR Propagation(64, 64)	FC(64, C)
10-layer GPR-GNN	Drop, FC(d, 64), ReLU, Drop, FC(64, 64), 10-layer GPR Propagation(64, 64)	FC(64, C)
10-layer GPR-GNN_2	Drop, FC(d, 16), ReLU, Drop, FC(16, 64), 10-layer GPR Propagation(64, 64)	FC(64, C)
2-layer GraphSAGE	SAGEConv(d, 64), Drop, ReLU, SAGEConv(64, 64)	FC(64, C)

Table 2: **Architecture details**. GATConv, GCNConv, and SAGEConv represent the convolutional layer of GCN, GAT, and GraphSAGE. Drop is the Dropout layer with a dropout rate of 0.5, d is the node feature dimension, and C is the number of classes. "(α, β)" represents that a module's input dimension is α, while the output dimension is β. Please see Sec B.3 for more details.

Group	Client ID	Model
3 clients	1	2-layer GAT
	2	2-layer GCN
	3	10-layer GPR-GNN
5 Clients	1	2-layer GAT
	2	3-layer GCN
	3	2-layer GCN
	4	10-layer GPR-GNN
	5	10-layer GPR-GNN
10 Clients	1	1-layer GAT
	2	2-layer GAT
	3	3-layer GAT
	4	1-layer GCN
	5	2-layer GCN
	6	3-layer GCN
	7	10-layer GPR-GNN
	8	10-layer GPR-GNN
	9	8-layer GPR-GNN
	10	8-layer GPR-GNN

Table 3: **Configurations for local models in the main experiment** under different total numbers of clients. Each model corresponds to the architecture in the Table 2. Please see Sec B.3 for more details.

Group	Client ID	Model
MHGNN_1	1	2-layer GraphSAGE
	2	2-layer GCN_2
	3	10-layer GPR-GNN_2
MHGNN_2	1	2-layer GraphSAGE
	2	2-layer GCN_2
	3	2-layer GAT_2
MHGNN_3	1	1-layer GAT
	2	2-layer GAT
	3	3-layer GAT
	4	1-layer GCN
	5	2-layer GCN
	6	10-layer GPR-GNN
	7	10-layer GPR-GNN

Table 4: **Configurations for local models employed in our supplementary experiments**. Each model corresponds to the architecture in the Table 2. Please see Sec C.1 for more details.

C Additional Experimental Results

In this section, we provide additional experimental results on the robust analyses of different local model configurations in Subsection C.1; varying local update epochs in Subsection C.2. We also provide a study to validate the effectiveness of prototype-based prediction and prediction ensemble operation in Subsection C.3.

C.1 Results on Varying Local Models

In this subsection, we run experiments under different local model architecture setups to validate the robustness of the proposed FedPPN. As shown in Table 4, we consider three sets of local model configurations, labeled as MHGNN_1, MHGNN_2, and MHGNN_3. Specifically, the models in these three model groups consist of GAT [Velickovic *et al.*, 2018], GCN [Kipf and Welling, 2017], and GraphSAGE [Hamilton *et al.*, 2017], with varying layers and hidden state dimensions. Here, we present the node classification accuracy of the three model groups in Table 5. Our FedPPN outperforms all baselines across the three additional local model configurations, especially in MHFL_3 by up to 15%.

C.2 Results on Varying Local Update Epochs

The number of local update epochs Q is an important hyperparameter for FL methods [Xie *et al.*, 2023]. Therefore, in this subsection, we study the impact of the different Q values for all baselines and the proposed FedPPN. Following [Wang *et al.*, 2022], we explore the node classification accuracy with different local update epochs $Q \in [1, 4, 16]$. Table 6 shows the result on the Cora dataset with the settings of five heterogeneous clients. From the results, we have two main observations. Firstly, the optimal selection of local update epochs Q varies for each method, and different Q values can signif-

Method	MHGNN_1	MHGNN_2	MHGNN_3
Local	77.47±0.3	75.11±0.7	56.29±0.8
FML	81.28±0.3	80.30±0.6	59.46±0.5
FedKD	77.18±0.9	76.82±0.1	56.83±0.5
FedProto	77.41±0.5	75.40±0.7	56.81±0.7
FedPCL	77.50±1.2	77.31±0.7	60.22±0.8
FedGH	75.30±1.7	75.10±1.7	56.41±1.2
FedPPN (ours)	**83.30±0.9**	**81.60±1.1**	**65.28±0.7**

Table 5: Experiments on varying local model setups. $\text{MHGNN}_i, i \in \{1, 2, 3\}$ represents different configurations for local models. Please see Sec C.1 for more details.

Method	$Q = 1$	$Q = 4$	$Q = 16$
Local	60.62±0.3	62.46±1.2	61.35±1.2
FML	62.52±3.3	68.93±1.7	64.59±1.1
FedKD	57.99±6.5	61.46±0.5	58.18±0.3
FedProto	61.38±0.6	63.34±0.7	62.48±1.5
FedPCL	67.83±0.4	67.20±1.3	67.04±1.2
FedGH	57.91±2.7	63.41±1.1	65.29±1.6
FedPPN (ours)	**77.37±1.5**	**77.61±1.6**	**75.94±2.7**

Table 6: Node classification accuracy by varying the local update epochs Q. Please see Sec C.2 for more details.

icantly impact accuracy. Secondly, FedPPN consistently outperforms other baseline models across various Q configurations, showing performance improvements of approximately 10% compared to the sub-optimal method. This result highlights our approach's robustness for local update epoch settings.

C.3 Effective Study on Prototype-based Prediction and Prediction Ensemble

In our FedPNN, we novelty propose combining prototype-based and local GNN predictions to obtain the final ensemble node predictions. In this subsection, we further explore the two questions: *Is the prototype-based prediction really valuable, and does our used ensemble operation really work?*

To answer the questions, we create line charts to visualize classification accuracy results of FedPPN: local GNN prediction, prototype-based prediction, and final ensemble prediction. Specifically, we show the visualization results on the Cora and CiteSeer datasets in Figure 1 for the setups of three total clients. The Figure shows that the prototype-based and ensemble predictions are significantly higher than the local GNN predictions across three datasets. This result demonstrates the effectiveness of the ensemble operation. Besides, it also provides strong proof to demonstrate the effectiveness of our proposed prototype propagation module.

References

[Blondel *et al.*, 2008] 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.

[Chien *et al.*, 2021] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive Universal Generalized PageRank Graph Neural Network. In *ICLR*, 2021.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *NeurIPS*, 2017.

[Huang *et al.*, 2023] Wenke Huang, Guancheng Wan, Mang Ye, and Bo Du. Federated graph semantic and structural learning. In *IJCAI*, 2023.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, pages 1–14, 2017.

[McMahan *et al.*, 2017] 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*, pages 1273–1282. PMLR, 2017.

[Namata *et al.*, 2012] 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, pages 1–8, 2012.

[Sen *et al.*, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective Classification in Network Data. *AI Mag.*, 29(3):93–106, 2008.

[Shen *et al.*, 2020] Tao Shen, Jie Zhang, Xinkang Jia, Fengda Zhang, Gang Huang, Pan Zhou, Kun Kuang, Fei Wu, and Chao Wu. Federated Mutual Learning. *arXiv preprint arXiv:2006.16765*, 2020.

[Tan *et al.*, 2022a] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. FedProto: Federated Prototype Learning across Heterogeneous Clients. In *AAAI*, volume 36, pages 8432–8440, 2022.

[Tan *et al.*, 2022b] Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. Federated Learning from Pre-Trained models: A Contrastive Learning Approach. *NeurIPS*, 35:19332–19344, 2022.

[Velickovic *et al.*, 2018] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *ICLR*, pages 1–12, 2018.

[Wang *et al.*, 2022] Zhen Wang, Weirui Kuang, Yuexiang Xie, Liuyi Yao, Yaliang Li, Bolin Ding, and Jingren Zhou. FederatedScope-GNN: Towards a Unified, Comprehensive and Efficient Package for Federated Graph Learning. In *KDD*, page 4110–4120, 2022.

[Wu *et al.*, 2022] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. Communication-efficient Federated Learning via Knowledge Distillation. *Nature communications*, 13(1):2032, 2022.

[Xie *et al.*, 2023] Yuexiang Xie, Zhen Wang, Dawei Gao, Daoyuan Chen, Liuyi Yao, Weirui Kuang, Yaliang Li, Bolin Ding, and Jingren Zhou. Federatedscope: A flexible federated learning platform for heterogeneity. 2023.

[Yao *et al.*, 2023] Yuhang Yao, Weizhao Jin, Srivatsan Ravi, and Carlee Joe-Wong. FedGCN: Convergence-communication tradeoffs in federated training of graph convolutional networks. In *NeurIPS*, 2023.

[Yi *et al.*, 2023] Liping Yi, Wang Gang, Xiaoguang Liu, Zhuan Shi, and Han Yu. FedGH: Heterogeneous Federated Learning with Generalized Global Header. In *ACM MM*, 2023.

[Zhang *et al.*, 2018] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep Mutual Learning. In *CVPR*, 2018.

[Zhang *et al.*, 2021] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. Subgraph Federated Learning with Missing Neighbor Generation. In *NeurIPS*, 2021.