Information Sciences 657 (2024) 119976

Contents lists available at ScienceDirect

Information Sciences

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

FedGL: Federated graph learning framework with global self-supervision

Chuan Chen [a], Ziyue Xu [a], Weibo Hu [a], Zibin Zheng [a,*], Jie Zhang [b]

[a] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[b] Tencent Inc., China

ARTICLE INFO

Keywords:
Federated learning
Graph learning
Graph neural networks
Self-supervised learning

ABSTRACT

Graph data are ubiquitous in the real world. Graph learning (GL) attempts to mine and analyze graph data so that valuable information can be discovered. Existing GL methods are designed for centralized scenarios. However, in practical scenarios, graph data are usually distributed across different organizations, resulting in isolated data silos. To address this problem, we incorporate federated learning into GL and propose a general Federated Graph Learning framework called FedGL. FedGL is capable of obtaining a high-quality global graph model while protecting data privacy by discovering the global self-supervision information during the federated training. Specifically, we propose uploading prediction results and node embeddings to the server for discovering the global pseudo labels and global pseudo graph. These are then distributed to each client to enrich the training labels and complement the graph structure respectively, thereby improving the quality of each local model. Moreover, the global self-supervision enables information of each client to flow and be shared in a privacy-preserving manner, thus alleviating the *heterogeneity* and utilizing the *complementarity* of graph data among different clients. Finally, experimental results show that FedGL significantly outperforms baselines on four widely used graph datasets.

1. Introduction

In the real world, graph data are ubiquitous, appearing in such as social networks, financial transaction networks, and biological networks. Graph learning (GL) aims to dig out the valuable information from the graph data by applying various graph models, including such as graph embedding [3], graph neural networks [39]. GL has boosted many applications, but existing GL methods are designed for a centralized learning scenarios, where graph data storage and model training are centralized. However, in most industries, graph data exists in the form of isolated islands [42] which distributed in different organizations or institutions. Considering a practical problem in the financial industry, each bank owns the customer information, transaction network, and default history. Banks have some common customers. There is a crucial need for banks to collaborate to conduct a comprehensive credit assessments of their customers and identify a common industry blacklist. An intuitive idea is to collect the graph data together and merge them into a large graph, and then feed it to existing GL methods. However, it is nearly impossible to collect such graph data

* Corresponding author.
E-mail addresses: chenchuan@mail.sysu.edu.cn (C. Chen), xuzy53@mail2.sysu.edu.cn (Z. Xu), huwb7@mail2.sysu.edu.cn (W. Hu), zhzibin@mail.sysu.edu.cn (Z. Zheng), fattyzhang@tencent.com (J. Zhang).

https://doi.org/10.1016/j.ins.2023.119976
Received 18 May 2021; Received in revised form 15 October 2023; Accepted 30 November 2023

Available online 5 December 2023

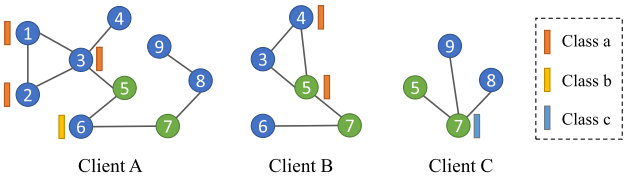

Fig. 1. *Heterogeneity* and *complementarity* in graph data among different clients is common in federated learning scenarios. As an example, Client A/B/C has 9/5/4 nodes, 9/5/3 edges, and 4/2/1 labels, respectively. The three clients vary in terms of the number of nodes, edges, and labels, and thus difference becomes more obvious on large-scale graphs in the real world. This is so-called the *heterogeneity*. On the other hand, there are overlapping nodes among different clients. In client A, nodes 1, 2, and 3 are closely connected and have the same label a, while nodes 4 and 5 are disconnected with node 5 having no label. However, in client B, nodes 4 and 5 are connected sharing label a. Therefore, by combining information from Client B, client A can infer that nodes 4 and 5 should also be closely connected and predicted to have label a. This is so-called the *complementarity*.

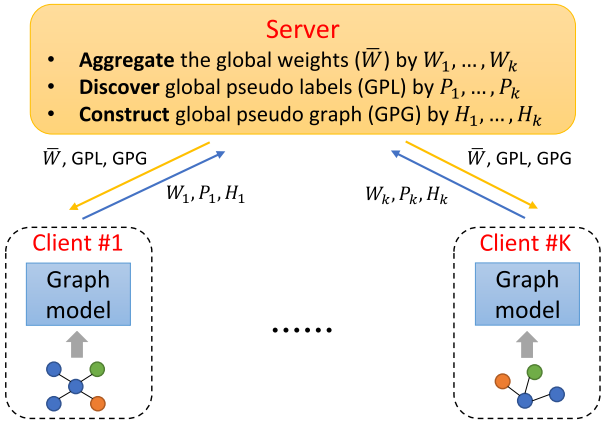

Fig. 2. The general framework of the proposed FedGL.

from institutions scattered nationally due to privacy and security concerns[1] and industry competition. Therefore, collaboratively training a high-quality graph model using graph data distributed across different organizations, without compromising data privacy, remains an open and crucial challenge.

Federated learning [26] is an emerging technique that trains machine learning models using datasets distributed across multiple devices, while preventing data leakage. The key idea is to keep the data on the devices (or clients) and collaboratively train a shared global model by uploading and aggregating the local updates (e.g., gradients or model parameters) yielded by clients to a central server. Commencing with the first and most famous federated learning algorithm FedAvg [26], many improved works have been proposed to address various problems of federated learning, including reducing the communication cost [18,32,22], overcoming the systems heterogeneity [30,27,44], overcoming the statistical heterogeneity [29,4,15], further protecting data privacy [9,1,21]. Intuitively, incorporating the federated learning framework into GL seems a promising solution for collaborative training without direct data sharing. However, existing research and applications are mainly focused on processing structured data, such as image and text data [46]. When our paper was originally composed, there are only few unpublished works [48,36] that attempt to develop a federated framework for graph data. [48] assumes that clients have the same nodes, different features and edges, and only one client has labels. [36] assumes that clients have the same nodes, features and edges, different labels. Since these works make different assumptions, they are difficult to generalize to address more varied federated GL problems. Moreover, in real-world scenarios, it is commonly observed that clients have different nodes, features, edges, and labels, with some overlapping nodes between clients. During the revision of this paper, there has been a resurgence of publications in the field of federated graph learning. [47] proposed a federated graph learning framework based on ego-graphs to tackle heterogeneity and utilize Mixup for a better privacy protection. [41] proposed leveraging graph contrastive learning to alleviate the performance drop resulting from differential privacy in federated graph learning. [37] proposed using a third-party server to conduct a privacy-preserving graph expansion protocol to incorporate high-order graph information into local model learning under privacy protection. Although these papers are related to the field of federated graph learning, their research focuses differ from ours. Specifically, they focus on the design of (more complex and advanced) privacy preservation strategies but do not fully explore the impact of topological incompleteness in federated heterogeneous graphs. This issue of topological incompleteness is one of the topics we will describe next.

In this paper, we explicitly point out two key challenges in federated graph learning. (1) *Heterogeneity*: Graph data distributed on different clients is essentially and potentially highly Non-Independent Identically Distributed (Non-IID). In this situation, the local

[1] On May 25, 2018, the European Union promulgated the EU General Data Protection Regulation (GDPR) to protect users' personal privacy and data security: https://gdpr-info.eu.

models trained on each client's graph data could have large differences. This leads to unsatisfactory global model performance after aggregation. (2) *Complementarity*: Graph data distributed on different clients usually contains complementary information due to the overlapping nodes. For these overlapping nodes, the graph structure on each client is incomplete owing to the inability to share and aggregate data. Fig. 1 illustrates the issues of *heterogeneity* and *complementarity*.

The above two challenges motivate the following goals during the federated training. (1) How can *heterogeneity* among graph data be mitigated, so that the server can aggregate models and obtain a high-quality global graph model? (2) How can the graph structure on each client complement one another to help clients learn better local graph models?

In this paper, we propose FedGL, a general Federated Graph Learning framework, which is capable of learning a high-quality graph model by discovering and exploiting the global self-supervision information to effectively deal with the *heterogeneity* and *complementarity*. It is worth noting that we originally introduce an solution that employs pseudo-labels and pseudo-graphs to tackle both challenges, a novel approach that has not been explored previously.

The general framework is shown in Fig. 2. There are K clients and one server. In FedGL, each client locally trains a graph model (local model) by using its graph data. Existing federated learning methods upload gradients or model parameters to the server which aggregates them to obtain a global model and distributes it in the next iteration. FedGL additionally upload the prediction results and node embeddings from each client to the server. This enables discovery of global self-supervision information, including global pseudo label and global pseudo graph, thereby alleviating *heterogeneity* and utilizing *complementarity*. Specifically, the global pseudo label is discovered by first fusing the prediction results, then selecting high confidence results for unlabeled nodes. Server distributes the discovered pseudo labels to each client, enriching local training labels and thereby improving the quality of local models. This global pseudo label discovery enables information flow and integration between clients in a privacy-preserving manner, thus mitigating *heterogeneity*. Additionally, we propose constructing a global pseudo graph by firstly fusing the node embeddings from each client, then reconstructing the full adjacent matrix. The Server distributes this constructed global pseudo graph to each client to complement local graph structure, thus further improving each local model and leading to a high-quality global model. The process of global pseudo graph discovery enables graph structures of each client to be collected and shared in a privacy-preserving manner, fully utilizing the *complementarity*.

As a general federated framework for the distributed graph data, FedGL is not restricted to any specific graph model. In this work, we adopt graph neural networks [39] as the graph model, which have demonstrated state-of-the-art performance on graph-based tasks. Extensive experiments conducted on four widely-used graph datasets show FedGL significantly outperforms centralized, simple federated and local methods, which fully verifies the effectiveness of FedGL.

Our main contributions are summarized as follows:

- We propose a novel federated graph learning framework FedGL that enables collaborative training of high-quality graph neural networks using distributed graph data from different clients, while preserving data privacy. FedGL provides a practical solution for organizations to perform cooperative graph learning in real-world scenarios.
- To discover global self-supervision informations, we design FedGL to aggregate prediction results and node embeddings from clients at the server. The extracted global pseudo labels and reconstructed global pseudo graph will be distributed to clients to augment local training and enhance model generalization. Specifically, the pseudo labels expand training labels at each client, while the pseudo graph complements local graph structure. This unique design improves the performance of FedGL and leads to better local models.
- The proposed global pseudo label dexterously enables the privacy-preserving information flow and integration across clients. In particular, high-quality clients can assist low-quality clients through the shared global pseudo labels, thereby mitigating the *heterogeneity*. Similarly, the global pseudo graph subtly aggregates graph structures from all clients under privacy protection. This allows fully utilizing structural *complementarity* across clients.
- We choose the graph neural network as the graph model and conduct extensive experiments on four widely used graph datasets. Experimental results show that FedGL significantly outperforms the centralized, vanilla federated, and local methods, which fully verifies the effectiveness of FedGL. Furthermore, plentiful parameter and ablation experiments verify the stability and robustness of FedGL.

The rest of this paper is organized as the followings. In Section 2, we review the related work on federated learning, graph learning, and self-supervised learning. Section 3 introduces the theoretical knowledge of graph neural networks and federated learning. In Section 4, we detail the proposed framework FedGL. Extensive experimental results and analysis are presented in Section 5 followed by the conclusion and future work in Section 6.

2. Related work

2.1. Federated learning

Federated learning is an decentralized learning technique that enables collaborative training of models across multiple devices (aka. clients), with each device maintaining its own local model and data. It maintains a shared global model on a server by aggregating locally computed updates from clients [18,26] and well solves the data isolated island problem and protects data privacy. Specifically, each client trains a local model and computes the local update based on its data. The local updates are uploaded to a server that aggregates them to update the global model. The updated model is further distributed to each client for the next round

of training. This process is iteratively executed until the global model converges. For example, FedAvg [26], the most representative federated learning method, uploads the model parameters and averages them to obtain the global model.

To date, many improvement efforts have been devoted to address various problems in federated learning, including reducing the communication cost [18,32,22], overcoming the systems heterogeneity [30,27,44], overcoming the statistical heterogeneity [29,4,15], further protecting data privacy [9,1,21], etc. Federated learning has promising applications in finance, healthcare, mobile edge networks, and many other industries [42], where data cannot be directly aggregated for training machine learning models due to factors such as intellectual property rights, privacy protection, and data security.

2.2. Graph learning

Graph learning (GL) aims to mine and analyze graph data to obtain valuable information. Due to the complexity of graph data, it is often necessary to first transform it into structured data, thus graph embedding [3] that embeds each node into a low-dimensional dense vector (node embedding) remains the most important technique in GL. These node embeddings can be readily applied to various downstream tasks, such as node classification, node clustering and link prediction. Up to now, extensive graph embedding methods have been proposed, and graph neural networks (GNNs) [39], an emerging type of neural network model on graphs, have achieved state-of-the-art performance on various graph-based tasks. GNNs integrate graph topology, node attributes and neural network to jointly learn node embeddings. Meanwhile, downstream tasks and node labels are incorporated to the model for end-to-end training. As the most important branch of GNNs, graph convolutional networks (GCNs) borrow ideas from convolutional neural networks (CNNs) and redefine the convolution operation for graph data. The pioneering work on GCNs is proposed in [2], which defines graph convolution by introducing spectral filters from the perspective of graph signal processing. Since then, there have been increasing improvements, approximations and extensions on spectral-based GCNs. The most well-known one is the improved version proposed by [17], which simplifies the spectral graph convolution by only using first-order neighbors. By stacking multiple convolutional layers, this GCN can encode both graph structure and node features node classification task. However, spectral-based GCNs require the entire graph as inputs and cannot scale to large graphs. To address this, spatial-based GCNs have been proposed. GAT [34] defines graph convolution by directly aggregating neighbor information. Combined with sampling and subgraph training strategies, the computation efficiency can be improved effectively. Further improvements mainly focus on convolution function and mechanism [34,33], expressive power and depth of network [40,23], large-scale and training efficiency [5,6,13], robustness [49,10], etc.

2.3. Self-supervised learning

Self-supervised learning (SSL) originates from computer vision and aims to learn visual features from a large number of unlabeled images or videos without using manually annotated information [14]. In recent years, SSL has gradually been used in the field of graph data learning [45,31]. SSL can be roughly categorized into two types. One uses pretext tasks that usually do not require labels, allowing the model to be trained in an unsupervised manner. For image data, common pretext tasks include image rotation, clustering, restoration, etc [14]. For graph data, common pretext tasks include node clustering, link prediction, graph partitioning, etc [45]. Another is to discover pseudo labels [19,11] and treat them as real labels to train models. Pseudo labels can be constructed based on source data, or discovered from prediction results. SSL can be used alone or used as a pre-training step [14]. SSL also can be used as a regularization item to help the main task achieve better results [38,31].

3. Preliminaries

3.1. Notations

A graph with N nodes is denoted as $G = (V, E)$, where $V = \{v_1, ..., v_N\}$ is the node set and $E \subseteq V \times V$ is the edge set. Let $A \in \mathbb{R}^{n \times n}$ represent the adjacency matrix. Let $D \in \mathbb{R}^{N \times N}$ denote the degree matrix, which is a diagonal matrix with $D_{ii} = \sum_j A_{ij}$. For an attributed graph, $X = \{x_1, ..., x_N\}$ is the associated node feature matrix and $x_i \in \mathbb{R}^d$ denotes the feature vector of node v_i, where d is the feature dimensionality. The node labels are represented as one-hot matrix $Y \in \mathbb{R}^{N \times C}$, where C is the number of classes of the node, and $Y_{ij} = 1$ if node i belongs to class j, otherwise $Y_{ij} = 0$.

3.2. Graph neural networks

Although there are numerous variants of GNNs, in this paper, we mainly focus on the most general and representative one proposed in [17]. For this GCN, the convolutional layer and layer-wise propagation rule are defined as

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right), \tag{1}$$

where $\tilde{A} = A + I_N$ is the adjacency matrix with added self-connections. I_N is the identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, and $W^{(l)}$ is the trainable weight matrix of layer l. $\sigma(\dots)$ is an activation function such as ReLU. $H^{(l)}$ is the latent representation matrix of layer l and $H^{(0)} = X$, i.e., using the node feature matrix as input. It is worth noting that the information of H is continually propagated through the immediate neighbors. Following [17], we consider a two layer GCN model to obtain the final node embeddings:

$$H = \hat{A}\,\sigma\left(\hat{A}XW^{(0)}\right)W^{(1)}, \tag{2}$$

where $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. Then, by inputting H to the Softmax function, we can obtain the predicted class probability matrix:

$$P_{ij} = \frac{e^{H_{ij}}}{\sum_{j=1}^{C} e^{H_{ij}}}, \tag{3}$$

where P_{ij} indicates the probability of node i belonging to class j. For semi-supervised node classification task, we compute the cross-entropy loss over the labeled samples:

$$L_{\text{GCN}} = -\sum_{i \in V_L}\sum_{j=1}^{C} Y_{ij}\log P_{ij}, \tag{4}$$

where V_L is the set of labeled nodes, C is the number of classes, and Y is the one-hot label matrix. Up to now, the neural network weights $W = \{W^{(0)}, W^{(1)}\}$ can be updated by back-propagation with the goal of minimizing Eq. (4).

3.3. Federated learning framework

In the federated learning framework, there are primarily two types of entities: clients and server. Clients refer to the party that owns data, such as mobile edge devices or organizations. The model trained and stored on each client is termed the *local model*. Server aggregates the local models uploaded by each client to obtain a *global model*. Specifically, assume there are K clients. Let D_k denotes the data owned by the client k, and W_k denotes the local model trained on the client k's data. W_G denotes the global model on the server. The training process of federated learning can be summarized as the following steps:

1. *Initialization*. Server determines the training tasks, hyper-parameters, initial model parameters W_G^0, etc., and distributes them to each client.
2. *Local model training*. Based on the global model W_G^t in t-th iteration, that is, each client utilizes its local data for training and updates the local model parameters. For the client k, the update formula of the t-th iteration is

$$W_k^t = W_k^t - \alpha\nabla L_{D_k}(W_k^t), \tag{5}$$

   where α is the learning rate, and the updated local model parameters are uploaded to the server.
3. *Global model update*. Server aggregates the model parameters uploaded by clients to obtain the updated global model parameters W_G^{t+1}, and then sends them to each client. The commonly used weighted average aggregation formula is as follows:

$$W_G^{t+1} = \sum_{k=1}^{K}\frac{|D_k|}{\sum_{k=1}^{K}|D_k|}W_k^t, \tag{6}$$

   where $|D_k|$ represents the number of samples of the client k.

The entire progress will repeat steps 1 and 3 until the global model converges or reaches the maximum number of iterations. In real-world applications, the number of clients can be substantial. Thus, in step 2, we can randomly select or assign certain clients to participate in training, thereby reducing training time. Additionally, we can upload model gradients instead of model parameters in step 2. If so, server aggregates the gradients and performs gradient descent to update the global model in step 3. Once federated training completes, the final global model can be used for prediction. The expectation is that its performance on the test set can be equivalent to that of a model trained centrally on aggregated data from all clients.

4. Federated Graph Learning Framework (FedGL)

4.1. Problem definition

In this work, we aim to propose a federated graph learning framework that collaboratively trains a high-quality graph model using distributed graph data from multiple clients while preserving data privacy. Suppose there are K clients, the graph data owned by the client k is denoted as $G_k = (V_k, E_k)$, where the number of nodes is $N_k = |V_k|$. The adjacency matrix, node feature matrix, and label matrix are denoted as $A_k \in \mathbb{R}^{N_k \times N_k}$, $X_k \in \mathbb{R}^{N_k \times d}$, and $Y_k \in \mathbb{R}^{N_k \times C}$, respectively. The total number of nodes of K clients is denoted as $M = \sum_{i=k}^{K} N_k$. Note that the graphs of different clients can vary in terms of number of nodes, structure, and label distribution. However there are some nodes overlap between the local graphs, i.e., for any graph G_k, there exists $k \neq t$, such that $V_k \bigcap V_t \neq \emptyset$. This setting reflects real-world graph data distributions and is thus practical and rational. In this scenario, we have identified the following two goals:

- *Global goal*. The global model is expected to achieve strong performance on the global test set. The global testing set is stored on the server and jointly determined by all client. Meeting this global objective is the original motivation for federated learning -

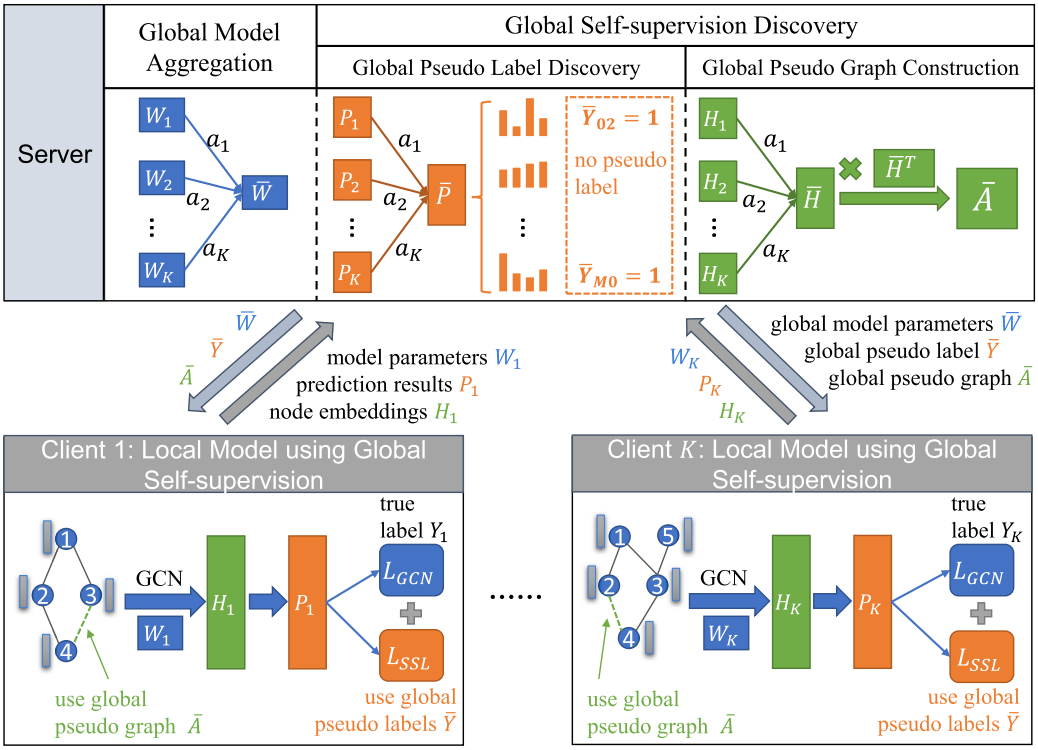

Fig. 3. The detailed framework of FedGL. L_{GCN} is the loss function of graph convolutional network, and L_{SSL} is the loss function of self-supervised learning. $a_1, ..., a_K$ is the aggregation weights. $\bar{Y}_{02} = 1$ indicates that the discovered global pseudo label of node 1 is class 3.

collaborating distributed client data to train a shared model that can approach the performance of a model trained centrally on aggregated data.

- *Local goal.* The global model should also demonstrate strong performance on each client's local test set. The local testing set refers to the testing set of each client's local data. This local objective entails that the global model outperforms models trained independently on each client's local data alone.

4.2. The framework of FedGL

Motivated by the objectives described above, we propose FedGL, a general Federated Graph Learning framework. The detailed framework is shown in Fig. 3. Overall, FedGL consists of two parts: 1) clients: local model training using global self-supervision, 2) server: global model aggregation and discovery of global self-supervision. The main ideas and workflow of FedGL can be summarized as:

- **Clients: local model training.** Each client trains a GCN model for several rounds on its local graph data, producing model parameters Wk W_k, node embeddings H_k, and prediction results P_k. These are uploaded to the server. Note that K clients train in parallel.
- **Server: global model aggregation.** Server performs weighted average aggregation on the local model parameters $W_1, ..., W_K$ to obtain the global model $\bar{W}$, and then distributes $\bar{W}$ to each client.
- **Server: global self-supervision discovery.** In addition to aggregating local model parameters, we propose discovering the global self-supervision information on the server, including global pseudo label and global pseudo graph, to address the *heterogeneity* and *complementarity*. Specifically, server first aggregates the prediction results $P_1, ..., P_K$ via weighted averaging to obtain the global prediction result $\bar{P}$. Server then selects the highest probability class from each row of $\bar{P}$ as the pseudo label for that node, forming the global pseudo label one-hot matrix $\bar{Y}$. Similarly, server aggregates node embeddings $H_1, ..., H_K$ using weighted averaging to obtain global node embedding $\bar{H}$. Multiplying $\bar{H}$ and its transpose reconstructs the full adjacency matrix, obtaining the weighted adjacency matrix $\bar{A}$ of the global pseudo graph. Server distributes the discovered global pseudo label $\bar{Y}$ and global pseudo graph $\bar{A}$ to each client for the next training round.
- **Clients: global self-supervision utilization.** Global pseudo labels act as "real" labels to enrich scarce real labels. This is done by constructing a self-supervised learning loss L_{SSL} and combining it with the main task loss L_{GCN} for joint optimization. The global pseudo graph is directly used to complement the incomplete graph structure. For example in Fig. 3, edge (3, 4) in client

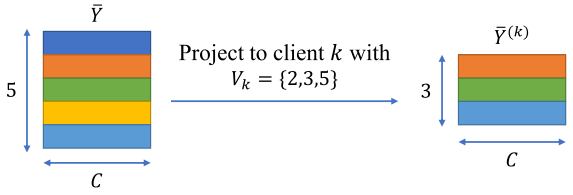

Fig. 4. Example of global pseudo label $\bar{Y}$ projection to client k.

1 and edge (2, 4) in client K have been well complemented. By exploiting the global pseudo labels and global pseudo graph, local model quality improves, leading to a high-quality global model.

The first two steps represent the standard federated learning processes, while the last two steps are the proposed global self-supervision discovery and utilization process, which are the core of FedGL. In summary, the global pseudo label dexterously enables client information flow and integration in a privacy-preserving manner. Specifically, high-quality clients can help low-quality clients through the pseudo labels, thereby mitigating the *heterogeneity*. Meanwhile, the global pseudo graph collects and shares graph structures privately to exploit *complementarity*. Furthermore, global pseudo label and global pseudo graph are complementary. Better node embeddings from the pseudo labels help construct a more accurate pseudo graph. In turn, an improved pseudo graph produces superior predictions and more reliable pseudo labels.

4.3. Local model using global self-supervision

For the local model of each client, we adopt the GCN model introduced in Section 3.2. And we improve the local models using the discovered global pseudo label and global pseudo graph.

4.3.1. Global pseudo label utilization

For a given client k, after receiving the global pseudo label $\bar{Y}$ which is represented as a one-hot matrix, it needs to project $\bar{Y}$ into its nodes to get $\bar{Y}^{(k)}$, since $\bar{Y}$ contains the pseudo label information of nodes on all clients. The projection process is illustrated in Fig. 4.

We then zero out all the rows of $\bar{Y}^{(k)}$ corresponding to the labeled nodes in the training set, keeping only rows for unlabeled nodes. The 1's in these rows represents the discovered pseudo label. Treating these as "real" labels, we augment the training set with them to participate in training. Concretely, we calculate a cross-entropy loss between the prediction result P_k and $\bar{Y}^{(k)}$ on the client k, which is called self-supervised learning (SSL) loss:

$$L_{SSL} = -\sum_{i \in V_{GPL}} \sum_{j=1}^{C} \bar{Y}_{ij}^{(k)} \log P_{k[ij]}, \tag{7}$$

where V_{GPL} denotes a set of unlabeled nodes with global pseudo labels, and $P_{k[ij]}$ denotes the i-th row and j-th column of the prediction result of the client k. By adding Eq. (7) to the GCN loss in Eq. (4), the final loss function of the local model on the client k is formulated as

$$\begin{aligned} L &= L_{\text{GCN}} + \alpha L_{\text{SSL}} \\ &= -\sum_{i \in V_L} \sum_{j=1}^{C} Y_{k[ij]} \log P_{k[ij]} - \alpha \sum_{i \in V_{PL}} \sum_{j=1}^{C} \bar{Y}_{ij}^{(k)} \log P_{k[ij]}, \end{aligned} \tag{8}$$

where α is a coefficient to control the strength of self-supervised learning. By introducing the SSL loss, the performance of main task can be effectively enhanced, which has been proved in many studies [19,38,31,45]. Moreover, in Eq. (8), the pseudo labels used for the SSL loss computation are the global pseudo labels, which is discovered by aggregating predictions from each client's local model. Thus, the global pseudo labels are believed to be more reliable than those obtained from a single client alone.

4.3.2. Global pseudo graph utilization

Taking the client k as an example. After receiving global pseudo graph $\bar{A}$ which is represented as a weighted adjacent matrix, it needs to project $\bar{A}$ into its nodes to obtain $\bar{A}^{(k)}$, since $\bar{A}$ contains the pseudo graph structure of nodes on all clients. The projection process is similar as Fig. 4. After obtaining the projection $\bar{A}^{(k)}$, it is easy to complement the original graph structure. The connection relationship that does not appear in the graph of client k can be complemented by the global pseudo graph. Existing connections in client k's graph can be further strengthened by the global pseudo graph. Implementation is straightforward - we simply fuse $\bar{A}^{((k)}$ with the original normalized graph structure. Corresponding to GCN model, we can directly update $\hat{A}$ in Eq. (2) as follows:

$$\hat{A}_k = \hat{A}_k + \beta \bar{D}^{-\frac{1}{2}} \bar{A}^{(k)} \bar{D}^{-\frac{1}{2}}, \tag{9}$$

where β is a coefficient to control the strength of the global pseudo graph to complement the graph structure, and $\bar{D}$ is the degree matrix of $\bar{A}^{(k)}$.

4.4. Global model

In Eq. (2), the trainable parameters of GCN model are $W = \{W^{(0)}, W^{(1)}\}$, where $W^{(0)}$ is related to the initial feature dimensionality of the nodes, and $W^{(1)}$ is related to the dimensionality of the hidden layer. Both of them are independent of the number of nodes. Hence, as long as the initial feature dimensionality and hidden layer dimensionality of each client are consistent with each other, server can directly aggregate the model parameters uploaded by different clients. Following FedAvg [26], we employ the weighted average aggregation method to aggregate the model parameters of K clients to obtain the global model:

$$\bar{W} = \sum_{k=1}^{K} \frac{N_k}{M} W_k, \tag{10}$$

where N_k is the number of nodes in the graph on the client k, and M is the sum of the number of nodes in the graph of the K clients, and W_k is the model parameters of the client k. $\frac{N_k}{M}$ denotes the proportion of the data volume of each client, which is used to measure the importance of its model parameters in aggregation. Intuitively, clients with larger amounts of training data are able to train better performing models, and thus their models could dominate during global model aggregation, i.e., be assigned larger weights. Furthermore, the weighting scheme can partially mitigate the impact of having an imbalanced data volume across clients.

Another thing worth mentioning here is that the choice of aggregation model does not affect the effectiveness of the self-supervised information presented in this article. Following the survey [35] and FedProx [20], we can also employ the aggregation method of FedProx to calculate the model parameters of the k-th client in the t-th iteration:

$$W_k^t \approx \arg\min_{W} \; L_k(W) + \frac{\mu}{2} ||W - \bar{W}^t||, \tag{11}$$

where L_k is the loss function (8), and μ is a coefficient that balances the original loss and the proximal term. FedProx [20] states that FedAVG is a special case of FedProx with $\mu = 0$. So the smaller mu is, the closer Fedprox is to FedAVG. The related comparative experiments will be given in Section 5.8.

4.5. Global self-supervision discovery

Eq. (10) introduces a weighting scheme based on the proportion of each client's data volume, which can partially mitigate the impact of imbalanced data volumes across clients. However, due to the *heterogeneity* in graph data between clients, including differences in graph structure and label distribution, the local models trained by different clients are often of uneven quality. This indicates that the global model obtained by weighted aggregation may still be unsatisfactory. Therefore, to obtain a high-quality global model, the fundamental challenge is to improve the performance of each local model by alleviating the *heterogeneity*. Additionally, the graph structure of each client contains complementary information due to overlapping nodes. Fully utilizing this *complementarity* information has the potential to further enhance the quality of local models.

In summary, the fundamental cause of the *heterogeneity* and *complementarity* across clients is that the graph data of each client cannot be centralized to train a unified model. Is there a privacy-preserving way to enable the information to flow and share between each client? In the federated learning framework, the server is naturally positioned to fulfill this role. As the raw data cannot be uploaded, in addition to the model parameters, other useful information can also be communicated to the server for aggregation and subsequent distribution to each client, thereby facilitating information flow. Therefore, we propose that clients additionally transmit prediction results and node embeddings to the server for discovery of global self-supervision information, including global pseudo labels and global pseudo graph. Server distributes these pseudo informations, augmenting local training labels and graph structure on each client. This aids in accounting for the *heterogeneity* and utilizing the *complementarity*.

4.5.1. Global pseudo label discovery

Upon receiving the prediction results uploaded by each client, server performs a weighted average fusion on the results, similar to (10), to obtain the global prediction results:

$$\bar{P} = \sum_{k=1}^{K} \frac{N_k}{M} P_k, \tag{12}$$

where $\frac{N_k}{M}$ is used as the fusion weight to measure the importance of the prediction results of the client k. Distinctly, a client with a larger amount of data possesses a more abundant graph structure, more extensive training labels, and potentially more accurate prediction results. Therefore, allocating greater weight to such clients can help ensure the fused global predictions are more accurate overall. Concurrently, the nodes with suboptimal predictions on certain clients with smaller datasets can also be improved through integration with the prediction outputs from other clients.

Utilizing $\bar{P}$, we aim to discover pseudo labels for self-supervised learning, an approach shown to be effective for image and graph data [19,38,31,11]. Specifically, we unearth high-confidence prediction results from $\bar{P}$ and obtain their predicted class labels, thus deriving global pseudo labels. For the prediction result vector $\bar{P}_i$ of the i-th node in $\bar{P}$, if its predicted probability for a given class exceeds a defined threshold, it is selected as a pseudo label for that node:

Algorithm 1 The algorithm of FedGL.

Input: Adjacent matrix $\{A_k\}_{k=1}^K$, node feature matrix $\{X_k\}_{k=1}^K$, label matrix $\{Y_k\}_{i=1}^K$, self-supervised learning coefficient α, global pseudo graph coefficient β, confidence threshold λ, neighbor number s.
Output: Global model $\bar{W}$

```
Server randomly initialize global model parameters W̄, and initialize global pseudo label Ȳ and global pseudo graph Ā to be zero matrices, distributing them to each client.
while not converge do
  // Client: local model training using global self-supervision
  for k = 1 to K do in parallel
    Use Ā to complement A_k by Eq. (9).
    Obtain the node embedding H_k and prediction result P_k by Eq. (2) and Eq. (3).
    Minimize the GCN loss and SSL loss in Eq. (8) to update the local model parameter W_k by back-propagation.
    Upload W_k, P_k, and H_k to the server.
  end
  // Server: global model aggregation and global self-supervision discovery
  Update global model W̄ by Eq. (10).
  Obtain global prediction results by Eq. (12).
  Discover the global pseudo label Ȳ by Eq. (13).
  Obtain global node embeddings by Eq. (14).
  Construct global pseudo graph Ā by Eq. (15).
  Distribute W̄, Ȳ, and Ā to each client.
end
```

$$\bar{Y}_{ij} = \begin{cases} 1, & \text{if } (\bar{P}_{ij} > \lambda) \text{ and } (j = \arg\max \bar{P}_i), \\ 0, & \text{otherwise}, \end{cases} \tag{13}$$

where $\bar{Y}$ is the one-hot matrix of the global pseudo label, and $\lambda \in [0, 1)$ is the confidence threshold for determining the pseudo label. A small value of λ allow the inclusion of marginally reliable predictions as pseudo-labels, yielding a relatively large number of pseudo-labels overall. That is, $\bar{Y}$ will contain more rows with 1's. Conversely, a large value of λ restricts pseudo-labels to only highly reliable predictions, resulting in a relatively small total number of pseudo-labels. That is, $\bar{Y}$ will contain more all-zero rows.

4.5.2. Global pseudo graph construction

The server performs weighted fusion on the node embeddings uploaded by each client, similar to (10), to obtain the global node embeddings:

$$\bar{H} = \sum_{k=1}^{K} \frac{N_k}{M} H_k, \tag{14}$$

where $\frac{N_k}{M}$ is also used as the fusion weight to measure the importance of the node embeddings of the client k. Recall that graph embedding maps each node into a low-dimensional dense vector while preserving topological structure information as much as possible. Tightly connected nodes in the original graph remain close to each other in the embedding vector space. Therefore, by calculating the distance or similarity between node vectors, the original graph structure can be approximately reconstructed. This idea has been commonly used in graph auto-encoder [16,28] or feature-based graph construction [24]. Based on this insight, we leverage the global node embeddings to construct the global pseudo graph:

$$\bar{A} = \phi(\bar{H}\bar{H}^T), \tag{15}$$

where $\phi(x) = \max(x, 0)$. As real-world graphs tend to be sparse, we limit the maximum number of neighbors of each node in the constructed pseudo graph to s. i.e., each row of $\bar{A}$ only reserves the largest s elements, and other elements are set to 0. Besides, the rows of $\bar{A}$ are normalized with $\sum_j^M \bar{A}_{ij} = 1$. Additionally, if the pseudo graph to be constructed is excessively large, large-scale graph construction methods may also be considered [24].

4.6. Model training

Algorithm 1 delineates the training process of FedGL, comprising client and server two parts. Clients train local models in parallel, while simultaneously using the global self-supervision information discovered by the server to improve the quality of local models. Server aggregates the uploaded local models to obtain the global model. More importantly, server discovers the global pseudo label and a global pseudo graph from the prediction results and node embeddings uploaded by clients. The clients and server proceed in an iterative alternating fashion until convergence of the global model.

5. Experiments

In this section, we conduct extensive experiments on the node classification task to empirically evaluate the effectiveness of the proposed framework FedGL. Additionally, experiments are conducted under different settings of federated learning. Finally, parameter study experiments are conducted to comprehensively analyze the developed FedGL framework.

Table 1
Statistics of datasets.

Dataset	#Nodes	#Edges	#Classes	#Features	#Training Labels	#Edge Sparsity	#Degree Distribution	#Label Sparsity
Cora	2708	5429	7	1433	140	0.074%	-	5.17%
Citeseer	3327	4732	6	3703	120	0.043%	0-degree much	3.61%
ACM	3025	13128	3	1870	60	0.143%	0-degree much	1.98%
Wiki	2405	17981	17	4973	340	0.311%	-	14.14%

In short, we conduct extensive experiments to answer the following questions:

- **Q1**: Whether FedGL can learn a high-quality global graph model, and its performance is close to or even better than the centralized method under the *global goal*?
- **Q2**: Whether the proposed global self-supervision can alleviate *heterogeneity* and utilize *complementarity* in graph data across clients to learn more superior node embeddings and achieve a better performance than the simple federated method under the *global goal*?
- **Q3**: Whether the learned global model can attain incremental performance gains compared with the local method under the *local goal*?
- **Q4**: Whether FedGL consistently performs well under different federated learning settings?
- **Q5**: How do the parameters of global self-supervision affect the performance of FedGL?
- **Q6**: Whether using another aggregation method affect the effectiveness of FedGL?

5.1. Datasets

Experiments are performed on four commonly utilized graph datasets including Cora, Citeseer, ACM, and Wiki [43,17,34,8,23]. Table 1 summarizes key statistics for these datasets. The calculation methods of the Table 1 headings are as follows:

- #Training Labels = 20 × #Classes (20 labels per class)
- #Edge Sparsity = #Edge / (#Nodes × #Nodes)
- #Degree Distribution: see Fig. 9 for specific data
- #Label Sparsity = #Training Labels / #Nodes

And the details of each dataset are as follows:

- **Cora**. It is an academic citation network, where nodes represent papers and edges represent citation relationships between papers. Paper fields are used as the node labels. The initial node features are derived from paper contents via bag-of-words representations.
- **Citeseer**. It is also an academic citation network. Like Cora, it uses bag-of-words representations as the initial node features.
- **ACM**. It is an academic collaboration network. Nodes represents papers, and edges represents co-author relationships. Papers are collected from three fields as node labels. The keywords of papers are transformed into bag-of-words representations as the initial node features.
- **Wiki**. It is a web-page link network constructed from english Wikipedia hyperlinks. Each node represents a web page containing an explanation of the term, and the edge represents the hyperlink references between web pages. Web page entry category is used as the node label. The content of web pages is transformed into bag-of-words representations as the initial node features.

In our experiments, the graph data are split into the training set, validation set, and testing set in two different ways to comprehensively evaluate the effectiveness of FedGL.

- *Fixed split*. Originating from [43], which uses all node features with 20 labels per class as the training set, 500 labels as the validation set for early-stopping, and 1000 labels as the testing set. This fixed split has been widely followed by the GCN related papers [17,34,8,23], since the split data is publicly available[2] and facilitates performance comparison between papers. If there are no special instructions in subsequent experiments, this split method will be adopted by default.
- *Random split*. It has more severely limited labels and greater randomness. For Cora, Citeseer, and Wiki, we randomly choose 5, 10, 15 labels per class as the training set, 500 labels for validation, and 1000 labels for testing. Since ACM only has 3 classes, we randomly choose 15, 25, 35 labels per class as the training set to ensure that the training labels are not too few so that the model can be learned normally.

[2] https://github.com/tkipf/gcn.

Table 2
Node classification accuracy under fixed split.

Dataset	Centralized	Federated	FedGL w/o GPG	FedGL w/o GPL	FedGL
Cora	0.811	0.810	0.828	0.812	**0.830**
Citeseer	0.705	0.676	0.732	0.676	**0.734**
ACM	0.848	0.855	**0.892**	0.858	0.891
Wiki	0.619	0.678	0.689	0.673	**0.691**

5.2. Comparison methods

Please note that there are very few studies focusing on graph data learning in federated scenarios. Several works that can be found are also under different scenario assumptions and cannot be directly compared. To demonstrate the rationality and effectiveness of FedGL, we compare with the following methods:

- **Centralized method (Centralized)**. For *global goal* comparison, the graph data (including training set, validation set, and testing set) of each client are collected and merged. The merged graph data are fed into the same GCN model for training. Finally, the trained model is evaluated on the global testing set. Note that this method is an ideal method that is not feasible in real scenarios, because it is often unrealistic to gather data together due to privacy security and industry competition.
- **Local method (Local)**. For *local goal* comparison, each client trains the same GCN model by feeding its graph data independently. Finally, the trained model is evaluated on the local testing set.
- **Simple federated method (Federated)**. For *global goal* and *local goal* comparison, this method uses the weighted average method to aggregate the local models to obtain the global model, which can be regarded as FedGL without global self-supervision. It is used to intuitively verify the effectiveness of the proposed global self-supervision module.
- **FedGL w/o GPG**. For *global goal* and *local goal* comparison, this method is an ablation version of FedGL by removing the global pseudo graph (GPG), which is used to verify the effectiveness of global pseudo graph.
- **FedGL w/o GPL**. For *global goal* and *local goal* comparison, this method is an ablation version of FedGL by removing the global pseudo label (GPL), which is used to verify the effectiveness of global pseudo label.

5.3. Experimental settings

5.3.1. Data settings for federated learning

To simulate the real-world graph data distribution, each client's data is derived via random sampling from the experimental datasets in varying different proportions. This ensures diversity across clients in terms of node number, graph structure, and label distribution, resulting in non-IID data while retaining some overlapping nodes due to the sampling process. As defined in Section 4.1, the federated scenario has two goals - the *global goal* and *local goal* - evaluated on the global and local test sets respectively. The specific implementation is as follows:

- *Global goal with global testing set*. The graph structure, feature matrix, and test labels on each client are merged to form the global test set. This is the most intuitive implementation to evaluate the global model, though the global test set can be customized as needed in practical settings. The final global model is evaluated on this global test set.
- *Local goal with local testing set*. The local test set for each client is simply the client's original test set. The final global model is distributed to each client and evaluated on the respective local test sets.

5.3.2. Parameter settings

FedGL consists of three modules: federated learning, GCN model, and global self-supervision. The GCN model parameters directly follow the settings of the original paper [17]: two convolutional layers, 16 hidden units, 0.5 dropout rate, 0.01 learning rate, and 5×10^{-4} L_2 regularization. For federated learning, we use 6 clients with the sampling proportions [30%,40%,50%,50%,60%,70%]. Additional parameters are: 100% client participation per round, 10 local training epochs per round, maximum 300 global rounds, and early stopping after 30 rounds without improvement. For global self-supervision, the confidence threshold λ is set to 0.1 for Wiki and 0.5 for other datasets. The self-supervised learning coefficient α is set to 0.1 for Wiki, 0.2 for other datasets. The global pseudo graph coefficient β is set to 1. The neighbor number s is set to 100. All the experiments are repeated 5 times and the average results are reported.

5.4. Q1Q2: Experimental results under global goal

5.4.1. Fixed split

Table 2 shows node classification accuracy under fixed split. The best results are highlighted in bold. As can be seen, FedGL remarkably outperforms both Centralized and Federated methods on all datasets. Further analysis yields the following observations:

- Compared to the ideal method Centralized, our proposed FedGL achieves approximately 2%-7% absolute performance improvement on various datasets. This shows that FedGL is not only unaffected by the inability to aggregate data, but also fully integrates

Table 3
Node classification accuracy under random split.

Dataset	Label ratio	Centralized	Federated	FedGL w/o GPG	FedGL w/o GPL	FedGL
Cora	5	0.549	0.587	0.629	0.580	0.640
	10	0.698	0.689	0.733	0.692	0.737
	15	0.740	0.738	0.799	0.739	0.806
Citeseer	5	0.555	0.577	0.610	0.579	0.605
	10	0.636	0.644	0.635	0.646	0.620
	15	0.648	0.662	0.708	0.662	0.710
ACM	15	0.723	0.754	0.811	0.760	**0.852**
	25	0.845	0.855	0.892	0.855	**0.892**
	35	0.902	0.902	**0.908**	0.901	0.903
Wiki	5	0.345	0.464	**0.484**	0.463	0.482
	10	0.435	0.516	**0.551**	0.510	0.550
	15	0.524	0.646	0.624	0.638	**0.651**

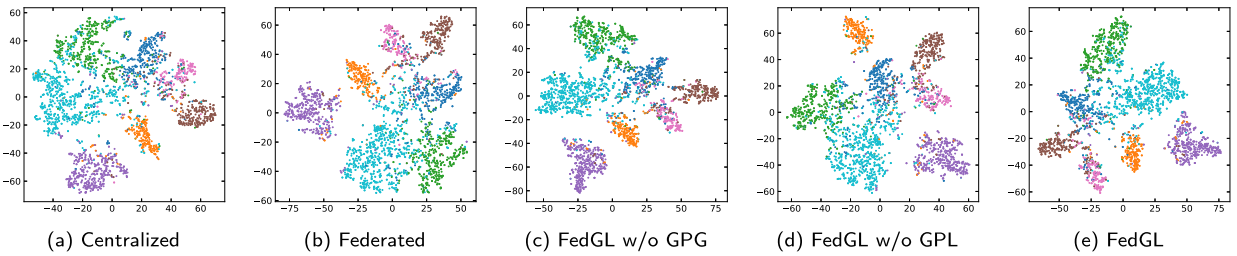

Fig. 5. Visualization of node embedding learned by different methods on Cora.

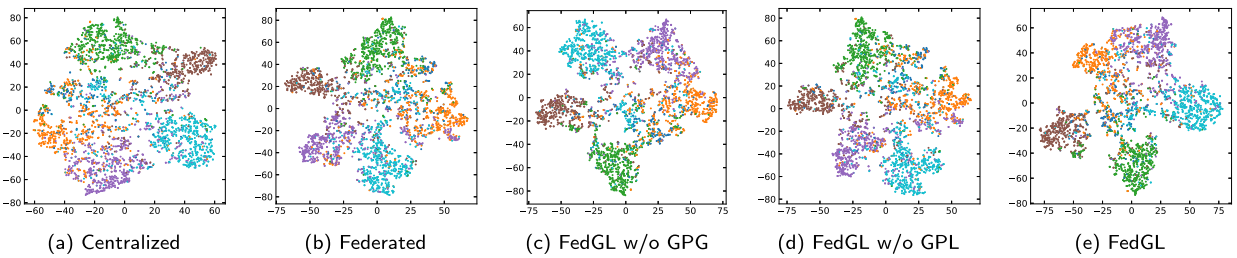

Fig. 6. Visualization of node embedding learned by different methods on Citeseer.

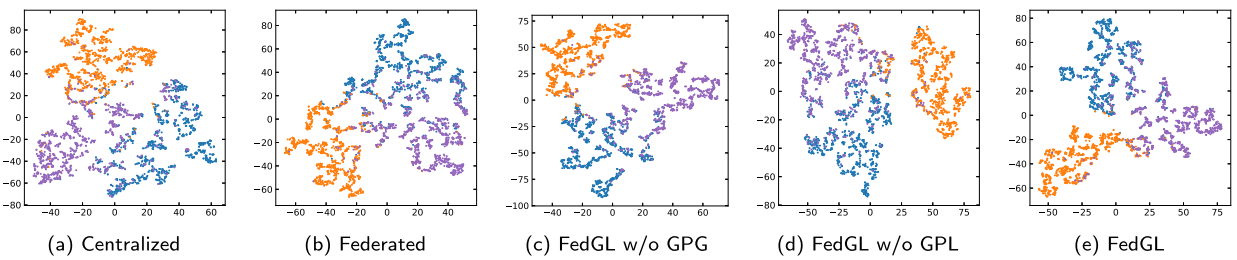

Fig. 7. Visualization of node embedding learned by different methods on ACM.

the data from each client for training, thereby learning a high-quality global model. FedGL outperforms Centralized for two reasons: (1) The proposed global self-supervision module improves each local model from the training labels and graph structure respectively, producing a high-quality global model. (2) Due to the particularity of graph data, there are overlapping nodes between clients, so the graph data of each client can be regarded as a sampling from the large graph data. Each client uses the sampled graph data to train a local model, which is equivalent to the process of Bagging ensemble learning, or understood as the process of data augmentation, so it is better than Centralized using merged single graph data.

- Compared to the Federated, FedGL outperforms on all datasets. Particularly on Citeseer, the performance improvement reaches 5.8%. Please note that the sole difference between Federated and FedGL is that the latter discovers and leverages global self-

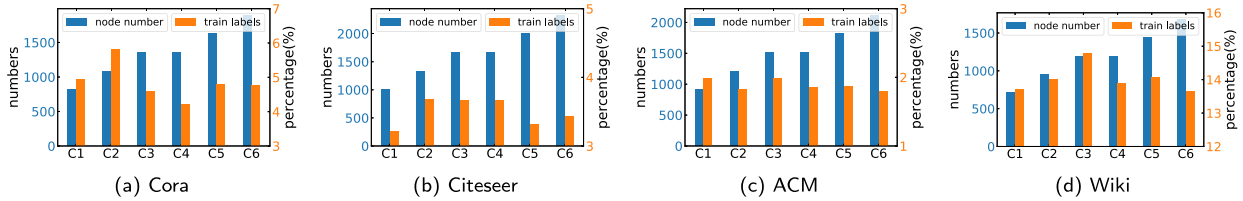

Fig. 8. The number of nodes and the proportion of training labels of each client (C1-C6) on different datasets.

supervision information to tackle the *heterogeneity* and *complementarity* across clients' graph data. This validates the efficacy of global self-supervision.

- In the ablation control group, we can see that FedGL w/o GPG surpasses the version w/o GPL and closely matches or slightly exceeds the complete FedGL. This indicates that the global pseudo label contributes more than the global pseudo graph to learn a high-quality global model. Meanwhile, the global pseudo label and global pseudo graph are essentially complementary to each other, so using both performs best overall.
- The results in Table 2 show FedGL achieves varying performance gains on different datasets. This can be attributed to differences in graph structure and label sparsity across datasets, which impact model effectiveness. As shown in Table 1, there are some obvious differences in edge sparsity, degree distribution, and label sparsity. Specifically, Citeseer and ACM have sparser graph structure and node labels. Such sparsity would be more severe under federated graph learning setting as each client only has a subset of the full data. Under such a situation, the proposed FedGL demonstrates greater advantages since the global pseudo label and global pseudo graph directly mitigate sparsity, improving local model quality and the global model. Hence, FedGL obtains higher performance improvement on Citeseer and ACM in Table 2. Furthermore, Table 3 shows FedGL achieves consistent and conspicuous improvements under label scarcity situation across datasets.

5.4.2. *Random split*

Considering potential model preferences under specific data splits, we introduce greater randomness and simulate strictly limited label scenario to repeat the node classification experiment. The experimental results are reported in Table 3. FedGL still dramatically outperforms Centralized and Federated on most datasets and label ratios, which further verifies the effectiveness of the proposed framework. Further analysis reveals the following observations:

- Compared with fixed 20-label per class splits, FedGL demonstrates greater advantages under random splits, particularly with very few training labels. When there are only 5 (ACM is 15) labels per class, FedGL obtains more than 10% absolute performance improvement compared to Centralized on Cora, ACM, and Wiki. Such a characteristic is especially suitable for practical applications since it is common to observe graphs with a small number of labeled nodes.
- Under various random split label ratios, FedGL consistently outperforms Federated, gaining over 5% absolute performance improvement in most cases, which fully verifies the stability and robustness of the proposed global self-supervision module.

5.4.3. *Visualization of node embedding*

Except for the performance comparison, we intuitively compare the quality of node embeddings by visualization. Concretely, we firstly feed the global testing set into the model learned by Centralized, Federated, FedGL w/o GPG, FedGL w/o GPL, and FedGL to obtain the node embeddings. Then, we map the embeddings into a 2-dimensional space with t-SNE algorithm [25] and draw a scatter plot. Fig. 5–7 is the visualization results of Cora, Citeseer, and ACM. Wiki is excluded as its 17 classes are difficult to distinguish by color. In the figures, each scattered point represents a node, and the node with the same color belongs to the same category. As observed, the nodes in Centralized are scattered with substantial inter-class overlaps. Federated has fewer overlapping nodes, but nodes remain dispersed without clear class boundaries. In contrast, the nodes in FedGL are quite compact, and the boundaries between classes are clear. This verifies FedGL can learn more discriminative node embeddings, explaining its superior performance over Centralized and Federated on downstream tasks

5.5. *Q3: Experimental results under local goal*

5.5.1. *Graph data distribution of clients*

To more intuitively understand the experimental results under the local goal, we visualize the number of nodes and the proportion of training labels for 6 clients on 4 datasets in Fig. 8. It can be seen that the proportion of training labels of each client is quite different. For example, in Cora, client 2 has the highest proportion of training labels, while client 3 and client 4 with the same number of nodes have different proportions of training labels. Fig. 9 shows the node degree distribution for 6 clients on 4 datasets. As can be seen, the node degree distribution of each client is roughly similar, because they are all randomly sampled from the original graph data, but there are also certain differences. For example, in all datasets, the node degree distribution of client 1 is obviously different from other clients. It has more 0-degree nodes and 1-degree nodes. In other words, it has more isolated nodes and nodes with only one edge, which indicates the graph structure of client 1 is very poor. In summary, the graph data between clients in the experiment are Non-IID, and there exists highly *heterogeneity* and *complementarity*.

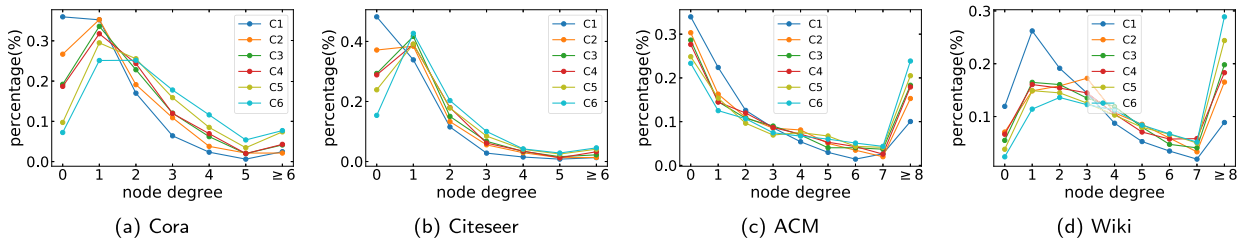

Fig. 9. The node degree distribution of each client (C1-C6) on different datasets.

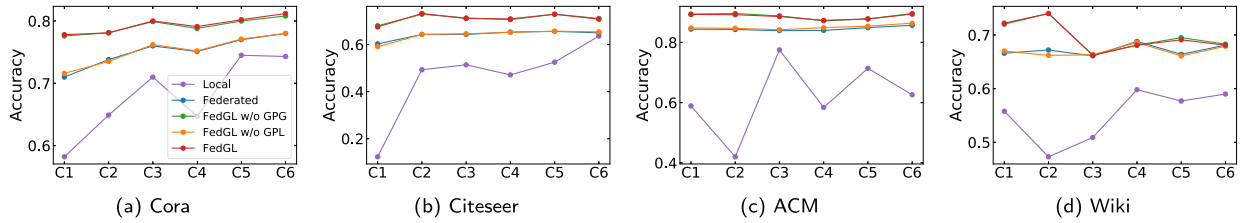

Fig. 10. Node classification accuracy of each client (C1-C6) on different datasets.

5.5.2. *Performance comparison*

Fig. 10 shows the node classification accuracy under local goal. FedGL significantly outperforms Local, and is also distinctly superior to Federated. Further analysis reveals the following observations:

- Compared with Local, FedGL achieves superior performance on every client and each dataset, with relatively stable results across clients despite their non-IID graph data. There are two main reasons: (1) Federated learning effectively cooperates with the data of each client for training, so that the learned global model performs better and more stable than the model that only uses the data of each client for training. (2) The proposed global self-supervision module discovers the useful information between clients and transmits it to each client through the server, thereby further improving the performance of the global model.
- Compared with Federated, FedGL introduces a global self-supervision module which enriches training labels using the global pseudo label and complements structure through the global pseudo graph. This directly improves the quality of each local model, leading to an improved global model. Therefore, FedGL can also perform better under the local goal.

5.6. *Q4: Different settings for federated learning*

5.6.1. *Client participation ratio per round*

In the above experiment, we use the settings in Section 5.3 by default. The client participation ratio per round is set to 100%, i.e., all clients participate in federated training in each round. In real scenarios, due to the large number of clients, or the differences in computing power and network bandwidth of each client, it is time-consuming if all clients are required to participate in each round. Therefore, in this experiment, we randomly select 30%, 50%, 70%, and 90% of the clients from 6 clients (i.e., 2, 3, 4, 5 clients) to participate in federated training in each round. The experimental results are reported in Table 4. There are two observations. (1) Compared to Table 2, FedGL has almost no performance decline under different client participation ratios, and the accuracy even has a slight improvement on Citeseer. This indicates FedGL can preserve accuracy while ensuring training efficiency for real-world applicability. (2) FedGL is still better than Federated, which manifests that the useful global self-supervision information can still be discovered to improve the quality of the global model, although the prediction results and node embeddings uploaded in each round are reduced.

5.6.2. *Number of clients and data size*

In this experiment, we aim to explore the impact of the number of clients and data size for federated learning. We change the sampling proportion of each client from [30%,40%, 50%,50%,60%,70%] to [30%,40%,50%], [20%,40%,60%], [50%, 60%,70%,80%], [20%,40%,50%,70%,70%,90%] with other parameters unchanged. The experimental results are reported in Table 5. There are three observations. (1) Under 4 groups of the different number of clients and data size, FedGL and Federated both achieve the best results in [50%,60%,70%,80%]. This observation shows that the data size of clients is more important than the number of clients. Particularly when clients have relatively large and size-balanced datasets, a higher-quality global model can be learned. (2) FedFL still outperforms Federated in most case, especially under [20%, 40%, 50%, 70%, 70%, 90%], which shows that the proposed global self-supervision module has effectively alleviated the *heterogeneity* of graph data between clients and learned a high-quality global model under relatively severe Non-IID situation. (3) Compared to Table 2, despite varying client counts and data sizes, FedGL maintains promising performance and its advantage.

Table 4
Node classification accuracy under different client participation ratio per round.

Dataset	Method	30%	50%	70%	90%
Cora	Federated	0.798	0.798	0.815	0.814
	FedGL	0.822	0.826	0.826	0.828
Citeseer	Federated	0.651	0.674	0.686	0.684
	FedGL	0.736	0.736	0.738	0.738
ACM	Federated	0.777	0.841	0.840	0.844
	FedGL	**0.862**	**0.886**	**0.868**	**0.888**
Wiki	Federated	0.668	0.684	0.686	0.686
	FedGL	**0.696**	**0.695**	**0.698**	**0.696**

Table 5
Node classification accuracy under different number of clients and data size. D1: [30%,40%,50%], D2: [20%,40%,60%], D3: [50%,60%,70%,80%], D4: [20%,40%,50%,70%,70%,90%].

Dataset	Method	D1	D2	D3	D4
Cora	Federated	0.778	0.794	0.808	0.810
	FedGL	0.816	0.817	0.838	0.824
Citeseer	Federated	0.690	0.692	0.706	0.702
	FedGL	0.730	0.692	0.741	0.747
ACM	Federated	**0.718**	**0.758**	0.875	0.849
	FedGL	0.609	0.625	**0.888**	**0.885**
Wiki	Federated	0.666	0.657	0.686	0.682
	FedGL	**0.671**	**0.684**	**0.701**	**0.694**

Table 6
Node classification accuracy under different overlapping node ratio between clients.

Dataset	Overlapping ratio	Centralized	Federated	FedGL w/o GPG	FedGL w/o GPL	FedGL
Cora	5%	0.797	0.800	0.829	0.795	0.828
	10%	0.800	0.802	0.828	0.805	0.817
	15%	0.794	0.790	0.820	0.794	0.818
Citeseer	5%	0.696	0.692	0.730	0.699	0.730
	10%	0.681	0.670	0.632	0.676	0.720
	15%	0.699	0.700	0.733	0.700	0.736
ACM	5%	0.870	0.870	0.870	0.870	**0.872**
	10%	0.716	0.708	**0.717**	0.708	0.712
	15%	0.890	0.891	0.891	0.892	**0.893**
Wiki	5%	0.691	0.690	0.694	0.690	**0.695**
	10%	0.664	0.659	0.666	**0.668**	0.664
	15%	0.680	0.675	0.677	**0.683**	0.674

5.6.3. Overlapping node ratio between clients

In this experiment, we explore the impact of the overlapping node ratio of the graph data between clients for federated learning. We keep the sampling proportion of the 6 clients unchanged while controlling their overlapping node ratios to 5%, 10%, 15%, and repeat the node classification experiment. The experimental results are reported in Table 6. There are two observations. (1) FedGL still outperforms Centralized and Federated, which proves the effectiveness and stability of the proposed global self-supervision. Besides, the two ablation versions of FedGL have achieved the best results in different datasets, which manifests that the global pseudo label and global pseudo graph also work well when used alone. (2) Compared to Table 2, the overlapping node ratio is about 1%. This experiment is 5%, 10%, and 15%. As can be seen, the performance is not positively correlated with the overlapping node ratio. Because the *heterogeneity* and *complementary* are opposite to each other to some extent. On the one hand, it is necessary to alleviate the *heterogeneity*. On the other hand, it is necessary to utilize the *complementarity*. This is exactly what FedGL focuses on and solves.

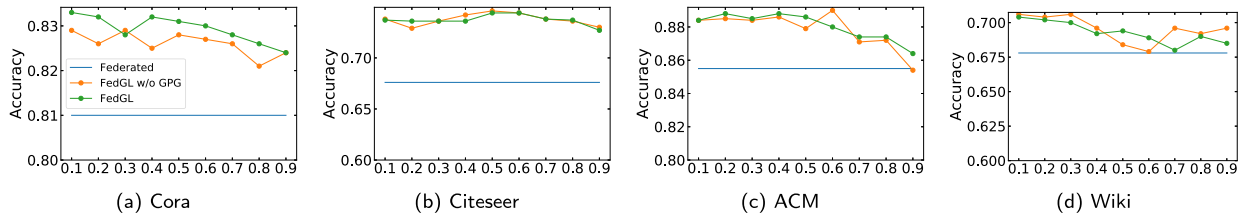

Fig. 11. Node classification accuracy under different confidence threshold λ.

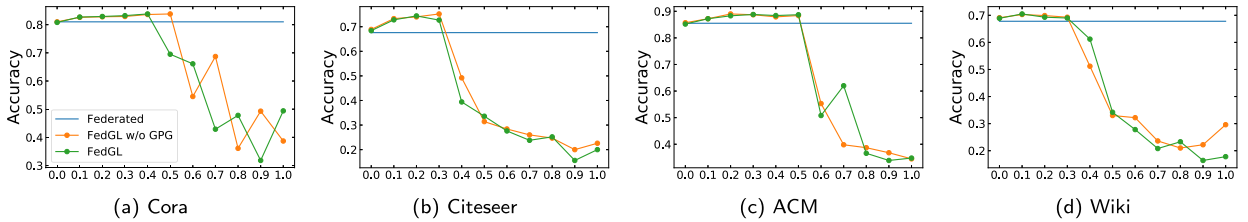

Fig. 12. Node classification accuracy under different self-supervision learning coefficient α.

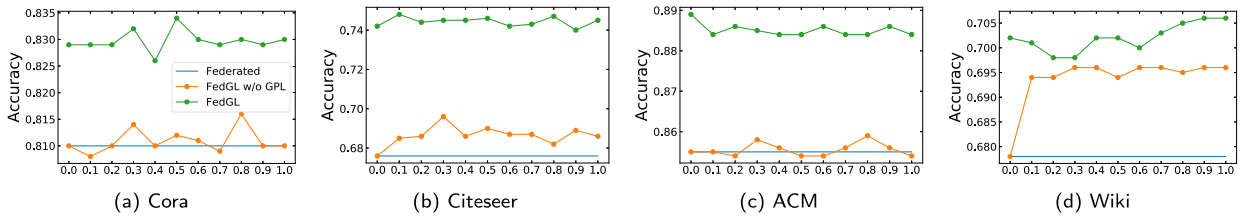

Fig. 13. Node classification accuracy under different global pseudo graph coefficient β.

5.7. Q5: Parameter sensitivity analysis

5.7.1. Confidence threshold

The confidence threshold λ in Eq. (13) controls the number of pseudo labels. To explore its impact on FedGL performance, λ is tuned from 0.1 to 0.9 in increments of 0.1. The results are shown in Fig. 11. On all datasets, FedGL and FedGL w/o GPG are significantly better than Federated under various λ, which shows the effectiveness and stability of the proposed global pseudo label. Notably, on Citeseer, classification accuracy of the global model improves by at least 5% when incorporating global pseudo labels. Optimal λ varies across datasets, but small values in the range [0.1, 0.3] are preferable overall.

5.7.2. Self-supervised learning coefficient

The self-supervised learning coefficient α in Eq. (8) controls the strength of self-supervised learning. To explore its impact, α is tuned from 0 to 1 in 0.1 increments. Note that $\alpha = 0$ means not using the global pseudo label, and $\alpha = 1$ means that the SSL loss is as important as the main task loss. The results are shown in Fig. 12. As α increases, the performance of FedGL and FedGL w/o GPG first increase, and then decrease sharply after exceeding a certain threshold. It is consistent with our analysis that the SSL item plays a supporting role to assist the main task, so α should not be set too large. With relatively small α, FedGL and FedGL w/o GPG both outperform Federated.

5.7.3. Global pseudo graph coefficient

The global pseudo graph coefficient β in Eq. (9) controls the strength of complementing the graph structure. To explore its impact, we tune β from 0 to 1 in increments of 0.1, where $\beta = 0$ omits the global pseudo graph and $\beta = 1$ weights it equally with the original graph structure. The results are shown in Fig. 13. As the value of β increases, the performance of FedGL and FedGL w/o GPG have some small fluctuations, but the overall performance is relatively stable and higher than Federated, which shows the effectiveness and stability of the proposed global pseudo graph.

5.8. Q6: Comparison of aggregation methods

In this section, we experimentally compared the effects of different aggregation methods on FedGL. The results Table 7 and Table 2 show the fact that FedGL mainly benefits from the proposed global self-supervision information, which is not affected by the change of aggregation method. Specifically, we used FedAVG [26] and FedProx [20] as the aggregation methods of FedGL and conducted comparative experiments. Further analysis reveals the following observations:

Table 7
Node classification accuracy under fixed split using FedProx.

Dataset	Centralized	Federated	FedGL w/o GPG	FedGL w/o GPL	FedGL
Cora	0.814	0.817	**0.837**	0.817	**0.837**
Citeseer	0.696	0.683	0.733	0.678	**0.737**
ACM	0.844	0.846	**0.894**	0.845	0.893
Wiki	0.620	0.673	**0.691**	0.678	0.686

Table 8
Wilcoxon signed-rank test p-value results including greater and two-sided (in parentheses) modes for node classification experiment under fixed split setting.

p-value	FedGL vs Comparison method (greater and two-sided modes, $\alpha = 0.05$)			
Dataset	Centralized	Federated	FedGL w/o GPG	FedGL w/o GPL
Cora	0.000976 (0.00195)	0.000976 (0.00195)	0.313(0.625)	0.000976 (0.00195)
Citeseer	0.000976 (0.00195)	0.000976 (0.00195)	0.188(0.375)	0.000976 (0.00195)
ACM	0.000976 (0.00195)	0.000976 (0.00195)	0.423(0.845)	0.000976 (0.00195)
Wiki	0.000976 (0.00195)	0.0801(0.160)	0.539(1.0)	0.116(0.232)

- Table 7 shows the node classification accuracy under fixed split using FedProx. The best results are highlighted in bold fonts. As can be seen, FedGL still has a great improvement compared with Federated and Centralized, which shows the effectiveness of our proposed global self-supervision information.
- The comparison between Table 2 and Table 7 shows that using FedProx [20] instead of FedAVG [26] does slightly improve the accuracy of FedGL on node classification task, but most of improvements are only around 0.01, which means little effect. And regardless of how much the model improves, the fact that changing the aggregation method does not make FedGL less effective shows that it does not depend on a specific aggregation method.

Therefore, the improvement brought by FedGL is mainly due to the proposed global self-supervision information, and this part is not affected by the change of the aggregation method.

5.9. Significance tests

In this subsection, we provide a comprehensive statistical significance analysis to evaluate whether there exist significant performance difference between FedGL and other models.

Due to specific requirements regarding data distribution and variance in parametric significance tests (such as t-test), we opted for more versatile non-parametric approaches. For experiments with larger sample sizes, we employed the widely-used Friedman Test, followed by Wilcoxon-Holm post-hoc analysis [12], to directly provide comparative results for multiple models across multiple datasets. For experiments with smaller sample sizes, we increased the sample size by running experiments with different random seeds and subsequently used the Wilcoxon signed-rank test [7] to compare two models on a single dataset. These non-parametric tests do not rely on assumptions of normal distribution or homogeneity of variances, which enhances their versatility and widespread applicability. We conducted non-parametric significance tests for our two primary experiments including (1) Node classification experiment under fixed split, and (2) Node classification experiment under random split.

For experiment (1), we employ the Wilcoxon signed-rank test ("two-sided" mode) to analyze whether the performance of FedGL exhibits significant differences compared to other methods, where the significance level α is set to 0.05. If p-value output from Wilcoxon signed-rank test is less than α then the null hypothesis that there is no significant differences between FedGL and the comparison method should be rejected, which means that our proposed method FedGL is significantly different from the comparison method. In order to increase the sample size of test, we execute each model 10 times with different random seeds and pair the results that share the same seed number for measurement. More specifically, we first randomly select 10 seeds, resulting in a sample size $n = 10$ for the Wilcoxon signed-rank test. We then run each model with an identical seed on a specific dataset to acquire accuracies for that particular seed setting. Finally, we pair FedGL's 10 average accuracies with those of the comparison method that share the same seed, as required for the input of the Wilcoxon signed-rank test. In addition to the "two-sided" mode, we also run the "greater" mode of the Wilcoxon signed-rank test. This allows us to evaluate not only significant differences between FedGL and other models, but also to assess whether FedGL's performance surpasses theirs. Given that the "greater" mode is more stringent than the "two-sided" test, we present the "two-sided" results within parentheses.

Table 8 shows the p-value results for node classification accuracy under fixed split. **In general, FedGL exhibits significant differences in performance compared to the baseline methods across the majority of datasets**. The specific analyses are as follows:

- Compared to the main comparison methods, FedGL significantly outperforms both the Centralized and Federated methods on Cora, Citeseer, and ACM, which is primarily attributed to FedGL's ability to handle heterogeneity and exploit complementary information;

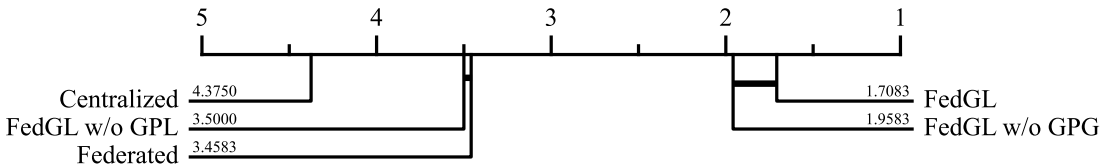

Fig. 14. Critical difference diagram for node classification experiment under random split setting.

- Compared to the ablation variants on Cora, Citeseer and ACM, FedGL significantly outperforms FedGL w/o GPL. Meanwhile, the performance of FedGL w/o GPG is comparable to FedGL;
- On Wiki, FedGL significantly outperforms the Centralized method, and shows no significant difference from the other three, which is consistent with the main experimental results in section 5.4.1. This can be attributed to the fact that the training effectiveness of the pseudo-labels and pseudo-graphs proposed in our paper is influenced by the graph structure of the dataset. A relatively sparse graph structure better highlights the effectiveness of FedGL.

And for experiment (2), owing to sufficient sample size, we directly adopt the prevalent Friedman Test followed by Wilcoxon-Holm Post-hoc Analysis to generate a critical difference diagram for significance testing, where the significance level α is set to 0.05 as well. Friedman Test can judge whether there exists significant difference among 5 methods, and Post-hoc analysis can further determine whether there are significant differences between each pair of models. The critical difference diagram serves as a visual representation of the test results. On the critical difference diagram, the x-axis illustrates the overall ranking of algorithms across multiple datasets, with models positioned to the right indicating superior performance. A bold horizontal line groups a set of classifiers that do not exhibit statistically significant differences.

Fig. 14 show the critical difference diagram [7] of node classification experiment under random split setting. We can clearly observe that FedGL and FedGL w/o GPG belong to one group, while Federated and FedGL w/o GPL methods belong to another group. The Centralized method stands alone as a separate group. The performances within one group do not exhibit significant differences. However, there are significant differences between different groups. The diagram indicates that our proposed FedGL method exhibits significant differences compared to the comparative methods Centralized and Federated. Moreover, FedGL is positioned at the far right of the x-axis and achieves the highest performance rank.

6. Conclusion

In this paper, we propose FedGL, a general federated graph learning framework which collaboratively trains a high-quality graph model across clients' data while preserving privacy. To address the *heterogeneity* and *complementarity* between clients' graph data, we propose discovering and utilizing global self-supervision information. Specifically, clients additionally upload prediction results and node embeddings to the server for discovering global pseudo label and global pseudo graph. Server then distributes them to each client to enrich the training labels and complement the graph structure. This enhances individual local models and produces a high-quality global model. More importantly, this global self-supervision discovery enables inter-client information flow and sharing in a privacy-preserving manner, thus mitigating the *heterogeneity* and utilizing the *complementarity*. Extensive experimentals on node classification task demonstrate that FedGL significantly outperforms the centralized, vanilla federated, and local methods, validating its efficacy.

Notably, FedGL is a general federated graph learning framework not restricted to specific graph models. In the future, we are interested in exploring the effectiveness of FedGL on more graph models such as GAT [34] and FastGCN [5]. We also aim to extend FedGL to scenarios where clients adopt different models and possess multimodal data.

CRediT authorship contribution statement

Chuan Chen: Conceptualization, Funding acquisition, Methodology, Supervision, Writing – original draft, Writing – review & editing. **Ziyue Xu:** Data curation, Formal analysis, Investigation, Software, Validation, Writing – review & editing. **Weibo Hu:** Data curation, Methodology, Software, Writing – original draft. **Zibin Zheng:** Funding acquisition, Project administration, Resources, Supervision. **Jie Zhang:** Resources, Validation.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgement

This work is supported by the National Key Research and Development Program of China (No. 2023YFB2703700), the National Natural Science Foundation of China (No. 62176269), the Guangzhou Science and Technology Program (No. 2023A04J0314) and the Tencent Wechat Rhino-bird project (No. 2021321).

References

[1] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, V. Shmatikov, How to backdoor federated learning, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 2938–2948.
[2] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, preprint, arXiv:1312.6203, 2013.
[3] H. Cai, V.W. Zheng, K.C.C. Chang, A comprehensive survey of graph embedding: problems, techniques, and applications, IEEE Trans. Knowl. Data Eng. 30 (2018) 1616–1637.
[4] F. Chen, M. Luo, Z. Dong, Z. Li, X. He, Federated meta-learning with fast convergence and efficient communication, preprint, arXiv:1802.07876, 2018.
[5] J. Chen, T. Ma, C. Xiao, Fastgcn: fast learning with graph convolutional networks via importance sampling, preprint, arXiv:1801.10247, 2018.
[6] W.L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, C.J. Hsieh, Cluster-gcn: an efficient algorithm for training deep and large graph convolutional networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 257–266.
[7] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[8] H. Gao, Z. Wang, S. Ji, Large-scale learnable graph convolutional networks, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2018, pp. 1416–1424.
[9] M. Hao, H. Li, G. Xu, S. Liu, H. Yang, Towards efficient and privacy-preserving federated deep learning, in: ICC 2019-2019 IEEE International Conference on Communications (ICC), IEEE, 2019, pp. 1–6.
[10] W. Hu, C. Chen, Y. Chang, Z. Zheng, Y. Du, Robust graph convolutional networks with directional graph adversarial training, Appl. Intell. (2021) 1573–7497, https://doi.org/10.1007/s10489-021-02272-y.
[11] W. Hu, C. Chen, F. Ye, Z. Zheng, Y. Du, Learning deep discriminative representations with pseudo supervision for image clustering, Inf. Sci. 568 (2021) 199–215.
[12] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.A. Muller, Deep learning for time series classification: a review, Data Min. Knowl. Discov. 33 (2019) 917–963.
[13] Z. Jia, S. Lin, R. Ying, J. You, J. Leskovec, A. Aiken, Redundancy-free computation for graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 997–1005.
[14] L. Jing, Y. Tian, Self-supervised visual feature learning with deep neural networks: a survey, IEEE Trans. Pattern Anal. Mach. Intell. (2020).
[15] M. Khodak, M.F. Balcan, A. Talwalkar, Adaptive gradient-based meta-learning methods, preprint, arXiv:1906.02717, 2019.
[16] T.N. Kipf, M. Welling, Variational graph auto-encoders, in: NIPS Workshop on Bayesian Deep Learning, 2016.
[17] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations (ICLR), 2017.
[18] J. Konečnỳ, H.B. McMahan, F.X. Yu, P. Richtárik, A.T. Suresh, D. Bacon, Federated learning: strategies for improving communication efficiency, preprint, arXiv:1610.05492, 2016.
[19] D.H. Lee, Pseudo-label: the simple and efficient semi-supervised learning method for deep neural networks, in: Workshop on Challenges in Representation Learning, ICML, 2013.
[20] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, preprint, arXiv:1812.06127, 2018.
[21] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, Q. Yan, A blockchain-based decentralized federated learning framework with committee consensus, IEEE Netw. (2020).
[22] L. Liu, J. Zhang, S. Song, K.B. Letaief, Edge-assisted hierarchical federated learning with non-iid data, preprint, arXiv:1905.06641, 2019.
[23] M. Liu, H. Gao, S. Ji, Towards deeper graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 338–348.
[24] W. Liu, J. He, S.F. Chang, Large graph construction for scalable semi-supervised learning, in: ICML, 2010.
[25] L. Van der Maaten, G. Hinton, Visualizing data using t-sne, J. Mach. Learn. Res. 9 (2008).
[26] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.
[27] T. Nishio, R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in: ICC 2019-2019 IEEE International Conference on Communications (ICC), IEEE, 2019, pp. 1–7.
[28] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, C. Zhang, Adversarially regularized graph autoencoder for graph embedding, preprint, arXiv:1802.04407, 2018.
[29] V. Smith, C.K. Chiang, M. Sanjabi, A. Talwalkar, Federated multi-task learning, preprint, arXiv:1705.10467, 2017.
[30] M.R. Sprague, A. Jalalirad, M. Scavuzzo, C. Capota, M. Neun, L. Do, M. Kopp, Asynchronous federated learning for geospatial applications, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2018, pp. 21–28.
[31] K. Sun, Z. Lin, Z. Zhu, Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 5892–5899.
[32] Z. Tao, Q. Li, esgd: communication efficient distributed deep learning on the edge, in: USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18), 2018.
[33] K.K. Thekumparampil, C. Wang, S. Oh, L.J. Li, Attention-based graph neural network for semi-supervised learning, preprint, arXiv:1803.03735, 2018.
[34] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, preprint, arXiv:1710.10903, 2017.
[35] O.A. Wahab, A. Mourad, H. Otrok, T. Taleb, Federated machine learning: survey, multi-level classification, desirable criteria and future directions in communication and networking systems, IEEE Commun. Surv. Tutor. 23 (2021) 1342–1397.
[36] B. Wang, A. Li, H. Li, Y. Chen, Graphfl: a federated learning framework for semi-supervised node classification on graphs, preprint, arXiv:2012.04187, 2020.
[37] C. Wu, F. Wu, L. Lyu, T. Qi, Y. Huang, X. Xie, A federated graph neural network framework for privacy-preserving personalization, Nat. Commun. 13 (2022) 3091.
[38] H. Wu, S. Prasad, Semi-supervised deep learning using pseudo labels for hyperspectral image classification, IEEE Trans. Image Process. 27 (2017) 1259–1270.
[39] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. (2020).
[40] K. Xu, C. Li, Y. Tian, T. Sonobe, K.i. Kawarabayashi, S. Jegelka, Representation learning on graphs with jumping knowledge networks, in: International Conference on Machine Learning, 2018, pp. 5449–5458.
[41] H. Yang, X. Zhao, M. Li, H. Chen, G. Xu, Mitigating the performance sacrifice in dp-satisfied federated settings through graph contrastive learning, Inf. Sci. (2023) 119552.
[42] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: concept and applications, ACM Trans. Intell. Syst. Technol. 10 (2019) 1–19.
[43] Z. Yang, W. Cohen, R. Salakhudinov, Revisiting semi-supervised learning with graph embeddings, in: International Conference on Machine Learning, PMLR, 2016, pp. 40–48.

[44] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, R. Yonetani, Hybrid-fl for wireless networks: cooperative learning mechanism using non-iid data, in: ICC 2020-2020 IEEE International Conference on Communications (ICC), IEEE, 2020, pp. 1–7.
[45] Y. You, T. Chen, Z. Wang, Y. Shen, When does self-supervision help graph convolutional networks?, in: International Conference on Machine Learning, PMLR, 2020, pp. 10871–10880.
[46] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, Knowl.-Based Syst. 216 (2021) 106775.
[47] T. Zhang, C. Mai, Y. Chang, C. Chen, L. Shu, Z. Zheng, Fedego: privacy-preserving personalized federated graph learning with ego-graphs, ACM Trans. Knowl. Discov. Data (2022).
[48] J. Zhou, C. Chen, L. Zheng, X. Zheng, B. Wu, Z. Liu, L. Wang, Privacy-preserving graph neural network for node classification, preprint, arXiv:2005.11903, 2020.
[49] D. Zügner, S. Günnemann, Certifiable robustness of graph convolutional networks under structure perturbations, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1656–1665.