FedGL: Federated Graph Learning Framework with Global Self-Supervision

CHUAN CHEN and WEIBO HU, Sun Yat-sen University, China

ZIYUE XU, Xi'an Jiaotong University, China

ZIBIN ZHENG, Sun Yat-sen University, China

Graph data are ubiquitous in the real world. Graph learning (GL) tries 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 in different organizations, i.e., the curse of isolated data islands. To address this problem, we incorporate federated learning into GL and propose a general Federated Graph Learning framework FedGL, which 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. Concretely, we propose to upload the prediction results and node embeddings to the server for discovering the global pseudo label and global pseudo graph, which are 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 the information of each client to flow and share 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, 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 using various graph model, including graph regularization [41], graph embedding [3], graph neural networks [50], etc. GL has boosted various applications, such as community detection [8], personalized recommendation [9], and fraud detection [36]. Existing GL methods are designed for a centralized learning scenario, that is, centralized graph data storage and centralized model training. However, in most industries, graph data exists in the form of isolated islands [53], i.e., 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 demand that banks hope to collaborate to conduct a comprehensive credit assessment on 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 almost impossible to collect the graph

Authors' addresses: Chuan Chen, chenchuan@mail.sysu.edu.cn; Weibo Hu, huwb7@mail2.sysu.edu.cn, Sun Yat-sen University, Guangzhou, China; Ziyue Xu, 1245570597@qq.com, Xi'an Jiaotong University, Xian, China; Zibin Zheng, zhzibin@mail.sysu.edu.cn, Sun Yat-sen University, Guangzhou, China.
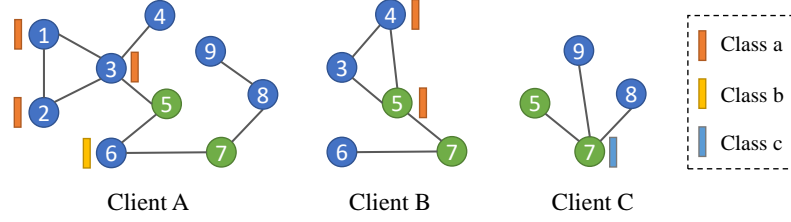
Fig. 1. An example of *heterogeneity* and *complementarity* of graph data between different clients in federated scenarios. Client A/B/C has 9/5/4 nodes, 9/5/3 edges, and 4/2/1 labels. The three clients vary in the number of nodes, edges, and labels, and this difference becomes more obvious on large-scale 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. Nodes 4 and 5 are not connected, and node 5 has no label. While in client B, nodes 4 and 5 are connected with the same label a. Therefore, client A can infer that nodes 4 and 5 should also be closely connected and predicted to have label a by combining the information of client B. This is so-called the *complementarity*.

data from institutions scattered around the country due to privacy security [1] and industry competition. Therefore, how to collaborate the graph data distributed on different organizations to train a high-quality graph model without compromising data privacy is an open and crucial problem.

Federated learning [34] is an emerging technique that trains machine learning models based on datasets distributed across multiple devices while preventing data leakage. The key idea is to leave the data on the devices (or clients) and 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 [34], many improved works have been proposed to address various problems of federated learning, including reducing the communication cost [23, 29, 44], overcoming the systems heterogeneity [35, 42, 55], overcoming the statistical heterogeneity [4, 20, 40], further protecting data privacy [1, 13, 26]. Intuitively, incorporating the framework of federated learning into GL is a promising solution for the above demand. However, existing federated learning related research and applications are mainly focused on processing structured data, such as image and text data [28, 53, 57], and very little work is focused on graph data. There are serval unpublished works [48, 59] that attempt to develop a federated framework for graph data. [59] assumes that clients have the same nodes, different features and edges, and only one client has labels. [48] assumes that clients have the same nodes, features and edges, different labels. Since they make different scenario assumptions, it is difficult to generalize them to address general federated GL problems. Besides, in the real world, it is commonly observed that clients have different nodes, features, edges, and labels, and have some overlapping nodes.

In general, there are two severe challenges for federated GL problems. (1) *Heterogeneity*: Graph data distributed on different clients are essentially and potentially highly Non-Independent Identically Distributed (Non-IID). In this situation, the local model trained by each client using its graph data could also has large differences, leading to an unsatisfactory global model after aggregation. (2) *Complementarity*: Graph data distributed on different clients usually contain complementary information due to the overlapping nodes. For these overlapping nodes, the graph structure on each client is not comprehensive due to the inability to share and aggregate data. Fig. 1 is an illustration of *heterogeneity* and *complementarity*.

---

[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

Server
- **Aggregate** the global weights ($\bar{W}$) by $W_1, \ldots, W_k$
- **Discover** global pseudo labels (GPL) by $P_1, \ldots, P_k$
- **Construct** global pseudo graph (GPG) by $H_1, \ldots, H_k$

$\bar{W}$, GPL, GPG $\bar{W}$, GPL, GPG

W_1, P_1, H_1 W_k, P_k, H_k

Client #1   **Client #K**

Graph model   Graph model

W: model parameters P: prediction results H: node embeddings
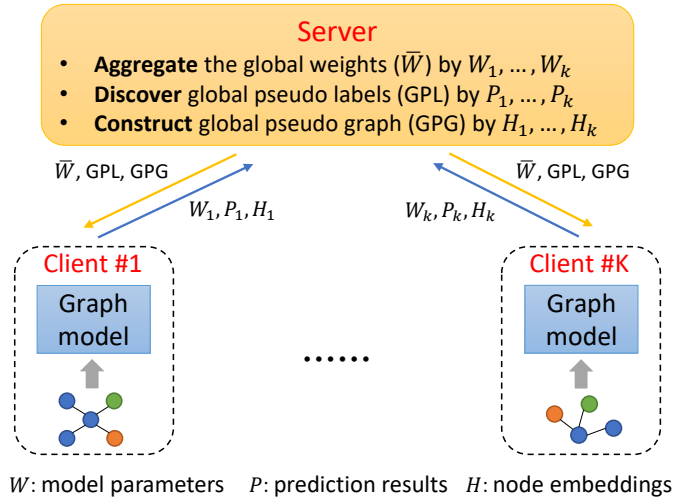
Fig. 2. The general framework of the proposed FedGL.

The above two challenges lead to two motivations during the federated training. (1) How to alleviate the *heterogeneity* of graph data between clients, so that server can aggregate and obtain a high-quality global graph model? (2) How to utilize the graph structure on each client to complement each other, so as to help each client learn a better local graph model?

In this paper, we propose a general <u>Fed</u>erated <u>G</u>raph <u>L</u>earning framework FedGL, 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*. The general framework is shown in Fig. 2. There are K clients and one server. Each client locally trains a graph model (local model) by using its graph data. Existing federated learning methods upload the gradients or model parameters to the server which aggregates them to obtain the global model and distributes it for the next iteration. For the proposed FedGL, we additionally upload the prediction results and node embeddings of each client to the server for discovering the global self-supervision information, including global pseudo label and global pseudo graph, thereby alleviating *heterogeneity* and utilizing *complementarity*. Concretely, we propose to discover the global pseudo label by firstly fusing the prediction results and then selecting the results of unlabeled nodes with high confidence. Server distributes the discovered pseudo label to each client to enrich the training labels, thereby improving the quality of each local model. The process of global pseudo label discovery enables the information of each client to flow and be integrated in a privacy-preserving manner, thus mitigating the *heterogeneity*. Besides, we propose to construct a global pseudo graph by firstly fusing the node embeddings from each client and then reconstructing the whole adjacent matrix. Server distributes the constructed global pseudo graph to each client to complement the graph structure, thus further improving each local model and leading to a high-quality global model. The process of global pseudo graph discovery enables the graph structure 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 [50] as the graph model, which have presented state-of-the-art performance on graph-based tasks. Finally, we conduct extensive experiments on four widely used graph datasets.

Experimental results show that FedGL significantly outperforms the centralized method, simple federated method, and local method, which fully verifies the effectiveness of FedGL.

Our main contributions are summarized as follows:

- We propose a general federated graph learning framework FedGL that can collaborate the graph data distributed on different clients to train a high-quality graph model while protecting data privacy. FedGL provides a feasible solution for cooperative graph learning between organizations in the real world.
- We propose to additionally upload the prediction results and node embeddings to the server for discovering the global self-supervision information, including global pseudo label and global pseudo graph, which are distributed to each client to enrich the training labels and complement the graph structure respectively, thereby improving the quality of each local model and obtaining a high-quality global model.
- The proposed global pseudo label dexterously enables the information of each client to flow and be integrated in a privacy-preserving way. Especially high-quality clients can give aid to low-quality clients through global pseudo labels, thereby alleviating the *heterogeneity*. The global pseudo graph subtly enables the graph structure of each client to be collected and shared in a privacy-preserving way, making full use of the *complementarity*.
- 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 method, simple federated method, and local method, which fully verifies the effectiveness of FedGL. Besides, 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 analyses 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 emerging decentralized learning technique that can collaboratively train multiple models with the training data distributed on different devices (also called clients) and maintain a shared global model on a server by aggregating locally computed updates [23, 34]. It 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 to continue the next round of training. This process is iteratively executed until the global model converges. For example, FedAvg [34], the most representative federated learning method, uploads the model parameters and averages them to obtain the global model. Up to now, many improvement efforts have been devoted to address various problems of federated learning, including reducing the communication cost [23, 29, 44], overcoming the systems heterogeneity [35, 42, 55], overcoming the statistical heterogeneity [4, 20, 40], further protecting data privacy [1, 13, 26], etc. Federated learning has a promising application in finance [33], healthcare [51], mobile edge networks [28], and many other industries [28], in which data cannot be directly aggregated for training machine learning models due to factors such as intellectual property rights, privacy protection, and data security. Besides, the framework of federated learning also has been effectively integrated into various techniques, such as multi-task learning [40], transfer learning [32], and reinforcement learning [60].

2.2 Graph Learning

Graph learning (GL) aims to mine and analyze the graph data, thus obtaining lots of valuable information. Due to the complexity of graph data, it is often necessary to firstly transform it into structured data, so graph embedding [3, 17] that embeds each node to a low-dimensional dense vector (node embedding) remains the most important technique of GL. These node embeddings can be readily applied to the downstream tasks, such as node classification, node clustering, link prediction, etc. Up to now, extensive graph embedding methods are proposed [58], such as the famous methods DeepWalk [38] and node2vec [11]. Recently, graph neural networks (GNNs) [50], an emerging type of neural network model on graphs, have presented state-of-the-art performance on various graph-based tasks. It integrates the graph topology, node attributes, and neural network to jointly learn node embeddings. Meanwhile, the downstream tasks and node labels are added 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 [39]. Since then, there have been increasing improvements, approximations, and extensions on spectral-based GCNs [7, 14, 22]. Among them, the most famous is the improved version proposed by [22], which simplifies the spectral graph convolution by only using the first-order neighbors. By stacking multiple convolutional layers, this GCN can encode both graph structure and node features to be useful for the node classification task. Since the spectral-based GCNs require the whole graph as the inputs and cannot scale to large graphs, spatial-based GCNs have been proposed, including GraphSAGE [12], GAT [47], and LGCN [10]. These methods define graph convolution via directly aggregating information from neighbors. By combining with sampling and subgraph training strategies, the computation efficiency can be improved effectively. After that, further improvements mainly focus on convolution function and mechanism [27, 45, 47], expressive power and depth of network [25, 30, 52], large-scale and training efficiency [5, 6, 18], robustness [15, 61], etc.

2.3 Self-Supervised Learning

Self-supervised learning (SSL) originates from the field of computer vision and aims to learn visual features from a large number of unlabeled images or videos without using any manually annotated information [19]. In recent years, SSL has gradually been used in the field of graph data learning [43, 56]. SSL can be roughly divided into two categories. One is to use pretext tasks, which usually do not require labels, thus the model can be trained in an unsupervised manner. For image data, common pretext tasks include image rotation, image clustering, image restoration, etc [19]. For graph data, common pretext tasks include node clustering, link prediction, graph partitioning, etc [56]. Another is to discover pseudo labels [16, 24] 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 [19]. SSL also can be used as a regularization item to help the main task achieve better results [43, 49].

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 [22]. 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 [22], we consider a two layer GCN model to obtain the final node embeddings:

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

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 mainly two types of entities, i.e., clients and server. Clients refer to the party that owns the data, which can be mobile edge devices or organizations. The model trained and stored on each client is called *local model*. Server aggregates the local models uploaded by each client to obtain a *global model*. Specifically, suppose there are K clients, D_k denotes the data owned by the client k, and W_k denotes the local model trained on the client k. 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 the 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 whole progress will repeat steps 2) and 3) until the global model converges or reaches the maximum number of iterations. In practical applications, the number of clients may be very large. Hence, in step 2), we can randomly select or designate some clients to participate in the training, thereby reducing the training time. In addition, we can also upload the 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). After the federated training is completed, the final global model can be used for prediction. It is expected that its performance on the testing set is equivalent to the model that collects data from clients for centralized training.

4 FEDERATED GRAPH LEARNING FRAMEWORK (FEDGL)

4.1 Problem Definition

In this work, we aim to propose a federated graph learning framework, which can collaborate the graph data distributed on different clients to train a high-quality graph model while protecting 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 there are certain differences in the number of nodes, graph structure, and label distribution of each client, but there are also some overlapping nodes between the clients. That is, for any graph G_k, there exists $k \neq t$, so that $V_k \bigcap V_t \neq \emptyset$. This setting is derived from the distribution of graph data in the real world, which is rational and practical. In such a scenario, we have clarified the following two goals:

- *Global goal.* It is expected that the global model achieves a promising performance on the global testing set. The global testing set is stored on the server and can be jointly determined by each client. The global goal is also the original intention of federated learning. It is expected that the data distributed on different clients could be collaborated to train a shared global model, whose performance can be close to the model that collects data from clients for centralized training.
- *Local goal.* It is expected that the global model also achieves a promising performance on the local testing set. The local testing set refers to the testing set of each client. Under the local goal, it is expected that the performance of the global model is better than the model that trains independently by only using the data of each client.

4.2 The Framework of FedGL

Based on the aforementioned motivations, we propose a general <u>Fed</u>erated <u>G</u>raph <u>L</u>earning framework FedGL. The detailed framework is shown in Fig. 3. Overall, FedGL consists of two parts: 1) clients: local model using global self-supervision, 2) server: global model aggregation and global self-supervision discovery. The main ideas and workflow of FedGL are summarized as follows:
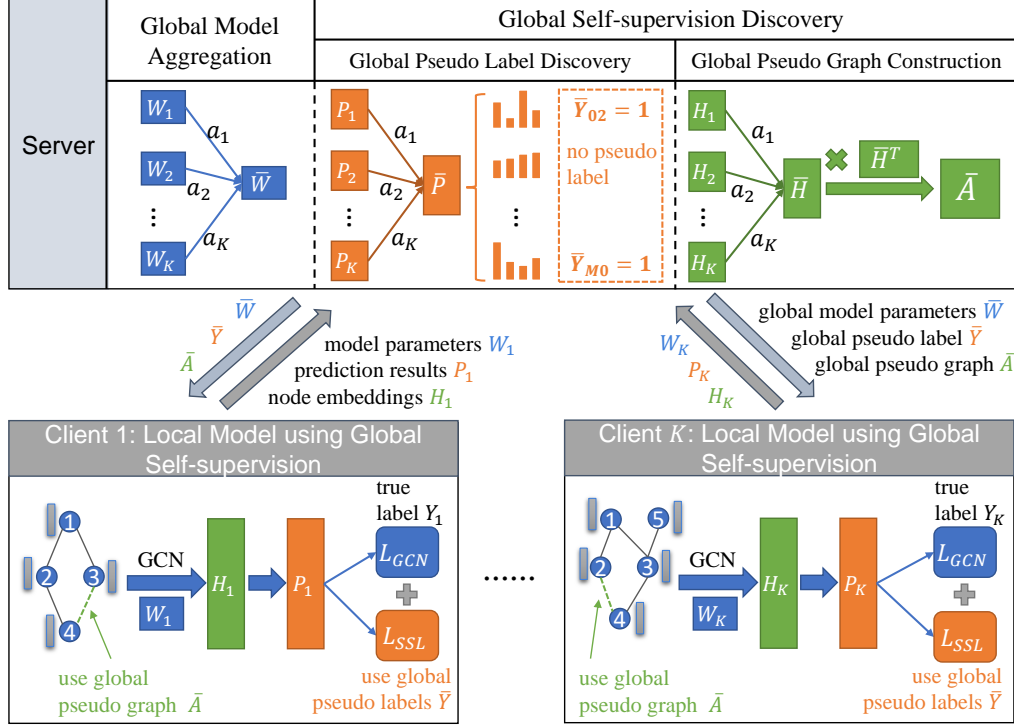
Fig. 3. The detailed framework of FedGL. L_{GCN} is the loss function of graph convolutional network, and L_{SSL} is the loss function of self-supervised learning. $a_1, ..., a_K$ is the aggregation weights. $\bar{Y}_{02} = 1$ indicates that the discovered global pseudo label of node 1 is class 3.

- **Clients: local model training.** Each client uses its local graph data to train several rounds of GCN model, obtaining model parameters W_k, node embeddings H_k, and prediction results P_k, then upload them to the server. Note that K clients train their local models in parallel.
- **Server: global model aggregation.** Server performs weighted average aggregation on the 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.** Except aggregating local model parameters to obtain a global model, we propose to discover the global self-supervision information on the server, including global pseudo label and global pseudo graph, to deal with the *heterogeneity* and *complementarity*. Specifically, server firstly performs a weighted average fusion on the prediction results $P_1, ..., P_K$ to obtain the global prediction result $\bar{P}$. Then, server selects the result with higher probability from the predicted probability vector of each row in $\bar{P}$ as the pseudo label of each node, which constitutes the one-hot matrix $\bar{Y}$ of the global pseudo label. Similarly, server performs weighted average fusion on the node embeddings $H_1, ..., H_K$ to obtain the global node embedding $\bar{H}$. By multiplying $\bar{H}$ and its transpose, server can reconstruct the whole 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 to start the next round of training.

- **Clients: global self-supervision utilization.** The global pseudo label is regarded as the "real" label to enrich the relatively rare real training labels by constructing a self-supervised learning loss L_{SSL} and adding it to 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 1 and edge (2, 4) in client K have been well complemented. By exploiting the global pseudo label and global pseudo graph, the quality of each local model can be effectively improved, thereby leading to a high-quality global model.

The above first two steps are 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, global pseudo label dexterously enables the information of each client to flow and be integrated in a privacy-preserving way, especially high-quality clients can give aid to low-quality clients through the global pseudo label, thereby alleviating the *heterogeneity*. The global pseudo graph subtly enables the graph structure of each client to be collected and shared in a privacy-preserving way, making full use of the *complementarity*. Meanwhile, global pseudo label and global pseudo graph are complementary to each other. Global pseudo label contributes to learning better node embeddings, thereby conducing to construct a more accurate pseudo graph, and global pseudo graph contributes to obtaining better prediction results, thus conducing to discover more reliable pseudo label.

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. At the same time, we use the discovered global pseudo label and global pseudo graph to improve the local model.

4.3.1 Global Pseudo Label Utilization. Taking the client k as an example, after receiving the global pseudo label $\bar{Y}$ that 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 shown in Fig. 4.
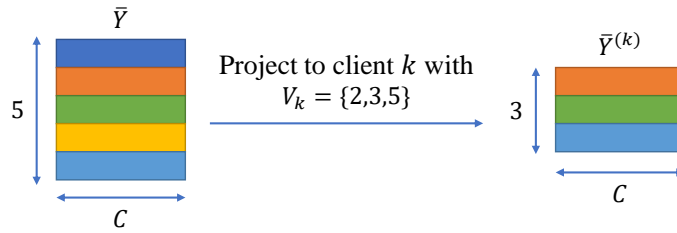

Fig. 4. Example of global pseudo label $\bar{Y}$ projection to client k.

We then set all the rows of $\bar{Y}^{(k)}$ corresponding to the labeled nodes in the training set to 0. That is, we only reserve the rows corresponding to the unlabeled nodes, in which 1 represents the discovered pseudo label. We regard these pseudo labels as the "real" labels and add them into the training set to participate in training. Concretely, we also 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

$$L = L_{\text{GCN}} + \alpha L_{\text{SSL}}$$

$$= -\sum_{i \in V_L} \sum_{j=1}^{C} Y_{k[ij]} \log P_{k[ij]} - \alpha \sum_{i \in V_{PL}} \sum_{j=1}^{C} \bar{Y}_{ij}^{(k)} \log P_{k[ij]}, \tag{8}$$

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 [24, 43, 49, 56]. Moreover, in Eq. (8), the pseudo label used to compute the SSL loss is global pseudo label, which is discovered by combining the prediction results of the local model on each client. Therefore, it is believed to be more reliable than the one discovered by the prediction results of only a single client.

4.3.2 Global Pseudo Graph Utilization. Taking the client k as an example, after receiving the global pseudo graph $\bar{A}$ that is represented as a weighted adjacent matrix, it also need to project $\bar{A}$ into its nodes to get $\bar{A}^{(k)}$, 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. The connection relationship that exists in the graph of client k can be further strengthened by the global pseudo graph. The specific implementation is also very convenient. We only need to 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 uncorrelated to the number of nodes. Therefore, 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 [34], 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, the larger the amount of data the client has, the better the model it trains, and then it should dominate during the aggregation, i.e., assigning a larger weight. Furthermore, the weighting way can reduce the impact of the imbalance of the data volume of each client to some extent.

4.5 Global Self-supervision Discovery

Eq. (10) introduces the proportion of data volume of each client as the aggregation weight to treat each client differentially, which can alleviate the impact of data imbalance between clients to some extent. However, due to the *heterogeneity* of graph data between clients, including graph structure and label distribution, the local models trained by different clients usually have uneven quality. It indicates that the global model obtained by weighted aggregation may still be unsatisfactory. Therefore, to obtain a high-quality global model, essentially, the quality of the local model on each client needs to be improved by alleviating the *heterogeneity*. On the other hand, the graph structure of each client is complementary, due to the overlapping nodes. Making full use of the *complementarity* is expected to further improve the quality of the local model.

In summary, the intrinsic reason for the *heterogeneity* and *complementarity* is that the graph data of each client cannot be collected to train a centralized model. Is there a privacy-preserving way to enable the information to flow and share between each client? In the framework of federated learning, server is naturally capable of accomplishing this task. Since the source data cannot be uploaded, in addition to the model parameters, other useful information can also be uploaded to the server for integration, and then distributed to each client, thus making the information flow. Therefore, we propose that clients additionally upload the prediction results and node embeddings to the server for discovering global self-supervision information, including global pseudo label and global pseudo graph. Server distributes them to each client to enrich the training labels and complement the graph structure, thereby alleviating the *heterogeneity* and utilizing the *complementarity*.

4.5.1 Global Pseudo Label Discovery. After receiving the prediction results uploaded by each client, similar as (10), server performs a weighted average fusion on the results to obtain the global prediction results:

$$\bar{P} = \sum_{k=1}^{K} \frac{N_k}{M} P_k, \tag{11}$$

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 has more abundant graph structure, more training labels, and more accurate prediction results. Therefore, assigning a larger weight can guarantee the fused global prediction results more accurate. At the same time, the nodes that have inferior predictions on some clients with a small amount of data can also become better by integrating the prediction results of other clients.

Based on $\bar{P}$, we try to discover pseudo labels for self-supervised learning, which has been proven to be effective in the learning of image and graph data [16, 24, 43, 49]. Concretely, we unearth these high-confidence prediction results from $\bar{P}$ and take out the predicted labels, thus obtaining the global pseudo labels. For the prediction result vector $\bar{P}_i$ of the i-th node in $\bar{P}$, if its predicted probability of a certain class is higher than a certain threshold, then it is selected as a pseudo label:

$$\bar{Y}_{ij} = \begin{cases} 1, & \text{if } (\bar{P}_{ij} > \lambda) \text{ and } (j = \arg\max \bar{P}_i), \\ 0, & \text{otherwise,} \end{cases} \tag{12}$$

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 means that a little more reliable prediction results could be selected as pseudo labels, so the number of pseudo labels is relatively large. That is, $\bar{Y}$ has more rows containing 1. A large value of λ means that

only enough reliable prediction results could be selected as pseudo labels, so the number of pseudo labels is relatively small. That is, $\bar{Y}$ has more rows with all 0s.

4.5.2   Global Pseudo Graph Construction. After receiving the node embeddings uploaded by each client, similar as (10), server performs a weighted fusion on the results to obtain the global node embeddings:

$$\bar{H} = \sum_{k=1}^{K} \frac{N_k}{M} H_k, \tag{13}$$

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 the idea of graph embedding, each node is mapped into a low-dimensional dense vector by preserving the topological structure information of the graph as much as possible. The original tightly connected nodes still keep close in the vector space. Hence, by computing the distance or similarity between node vectors, it can approximately reconstruct the original graph structure. This idea is also commonly used in graph auto-encoder [21, 37] or feature-based graph construction [31]. Based on this insight, we employ the global node embeddings to construct the global pseudo graph:

$$\bar{A} = \phi(\bar{H}\bar{H}^T), \tag{14}$$

where $\phi(x) = \max(x, 0)$. Since graphs in the real world are generally sparse, we limits the number of neighbors of each node in the constructed pseudo graph no more than 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 $\sum_j^M \bar{A}_{ij} = 1$. By the way, if the pseudo graph to be constructed is too large, any large-scale graph construction methods can also be considered [31].

4.6   Model Training

Algorithm 1 is the training process of FedGL. It mainly contains clients and server two parts. Clients are responsible for the training of local models in parallel. They simultaneously exploit the global self-supervision information discovered by the server to improve the quality of local models. Server is responsible for aggregating the local models uploaded by clients to obtain the global model. More importantly, server discovers the global pseudo label and global pseudo graph from the prediction results and node embeddings uploaded by clients. Clients and server alternately iteratively perform, until the global model converges.

5   EXPERIMENTS

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

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 mitigate the *heterogeneity* and utilize the *complementarity* of graph data between clients, so as to learn more superior node embeddings and achieve better performance than the simple federated method under the *global goal*?
- **Q3**: Whether the learned global model can gain some performance improvements compared to the local method under the *local goal*?

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 $\bar{W}$

1: Server randomly initialize global model parameters $\bar{W}$, and initialize global pseudo label $\bar{Y}$ and global pseudo graph $\bar{A}$ to be zero matrices, distributing them to each client.

2: **while** not converge **do**

3:    // *Client: local model training using global self-supervision*

4:    **for** $k = 1$ to K **do in parallel**

5:        Use $\bar{A}$ to complement A_k by Eq. (9).

6:        Obtain the node embedding H_k and prediction result P_k by Eq. (2) and Eq. (3).

7:        Minimize the GCN loss and SSL loss in Eq. (8) to update the local model parameter W_k by back-propagation.

8:        Upload W_k, P_k, and H_k to the server.

9:    **end**

10:    // *Server: global model aggregation and global self-supervision discovery*

11:    Update global model $\bar{W}$ by Eq. (10).

12:    Obtain global prediction results by Eq. (11).

13:    Discover the global pseudo label $\bar{Y}$ by Eq. (12).

14:    Obtain global node embeddings by Eq. (13).

15:    Construct global pseudo graph $\bar{A}$ by Eq. (14).

16:    Distribute $\bar{W}$, $\bar{Y}$, and $\bar{A}$ to each client.

17: **end**

Table 1. Statistics of datasets.

Dataset	#Nodes	#Edges	#Features	#Classes
Cora	2708	5429	1433	7
Citeseer	3327	4732	3703	6
ACM	3025	13128	1870	3
Wiki	2405	17981	4973	17

- **Q5**: Whether FedGL consistently performs well under different settings of federated learning?
- **Q5**: How do the parameters of global self-supervision affect the performance of FedGL?

5.1 Datasets

We conduct experiments on four widely used graph datasets [10, 22, 30, 47, 54], Cora, Citeseer, ACM, and Wiki. The statistics of these datasets are presented in Table 1. The details of each dataset are as follows:

- **Cora**. It is an academic citation network, each node represents a paper, and the edge represents the citation relationship between the papers. The field of the paper is used as the node label. The content of papers is transformed into bag-of-words representations as the initial node features.

- **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 network. Each node represents a paper, and the edge represents a co-author between the papers. The 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, derived from the English Wikipedia website. Each node represents a web page containing an explanation of the term, and the edge represents the hyperlink references between web pages. The category of the web page entry 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 [54], 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 [10, 22, 30, 47], 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.

5.2 Comparison Methods

Note that there are rare few studies focusing on graph data learning in federated scenarios. Several unpublished 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. In order to simulate the graph data distribution in the real world, the graph data of each client comes from the random sampling results of the experimental datasets under different proportions to

---

[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**

ensure the number of nodes, graph structure, and label distribution between clients to be diverse. That is, the graph data between clients are Non-IID. Meanwhile, there are some overlapping nodes between clients due to random sampling. In Section 4.1, we defined two goals in federated scenario, namely *global goal* and *local goal*, which are evaluated based on global testing set and local testing set. The following is the specific implementation:

- *Global goal with global testing set.* The graph structure, feature matrix, and testing labels on each client are merged as the global testing set. This is the most intuitive implementation to evaluate the global model. In practical scenarios, the global testing set can be specially customized. The final global model is evaluated on the global testing set.
- *Local goal with local testing set.* The local testing set is exactly the testing set of each client. The final global model is distributed to each client and evaluated on each local testing set.

5.3.2 Parameter Settings. FedGL consists of three modules, i.e., federated learning, GCN model, and global self-supervision. The parameters of the GCN model directly follow the settings of its original paper [22], i.e., 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 sampling proportion of each client [30%,40%,50%,50%,60%,70%], client participation ratio per round 100%, local training epochs of client per round 10, maximum iteration round 300, and early-stopping round 30. For global self-supervision, the confidence threshold λ is set to 0.1 for Wiki, 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 the node classification accuracy under fixed split. The best results are highlighted in bold fonts. As can be seen, FedGL remarkably outperforms Centralized and Federated on all datasets. Further analysis, we have the following observations.

- Compared with ideal method Centralized, FedGL gains about 2%-7% absolute performance improvement under various datasets, which indicates that FedGL is not only not affected by the inability to collect data, but also fully integrates the data of each client for training, and learns a high-quality global model. There are two reasons why FedGL outperforms Centralized. (1) The proposed global self-supervision module improves each local model from the training labels and graph structure respectively, leading to 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.

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**

- Compared with Federated, FedGL performs better under all datasets. Especially on Citeseer, the absolute performance improvement is up to 5.8%. Note that the only difference between Federated and FedGL is that FedGL discovers and exploits the global self-supervision information to tackle the *heterogeneity* and *complementarity* of graph data between clients. This result directly verifies the effectiveness of global self-supervision.
- FedGL w/o GPG achieves better performance than FedGL w/o GPL and is closer to or even slightly surpassing FedGL, which indicates that the global pseudo label is more helpful to learn a high-quality global model than the global pseudo graph. Meanwhile, the global pseudo label and the global pseudo graph are essentially complementary to each other, so using both simultaneously performs best in most cases.

5.4.2 Random Split. Considering that the model may exist data preferences for specific data split, we introduce greater randomness and simulate the strictly limited label scenario to repeat the above node classification experiment. The experimental results are reported in Table 3. FedGL still dramatically outperforms Centralized and Federated under most datasets and label ratios, which further verifies the effectiveness of the proposed framework. Further analysis, we have the following observations.

- Compared with fixed split (20 labels per class), FedGL shows more conspicuous superiority under random split especially when there are rare 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 label ratios of random split, FedGL consistently outperforms Federated, gaining more than 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 [46] and draw a scatter plot. Fig. 5-7 is the visualization results of
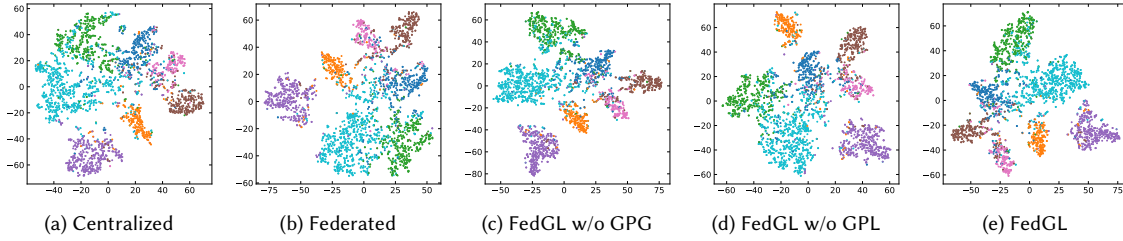
(a) Centralized   (b) Federated   (c) FedGL w/o GPG   (d) FedGL w/o GPL   (e) FedGL

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

(a) Centralized   (b) Federated   (c) FedGL w/o GPG   (d) FedGL w/o GPL   (e) FedGL
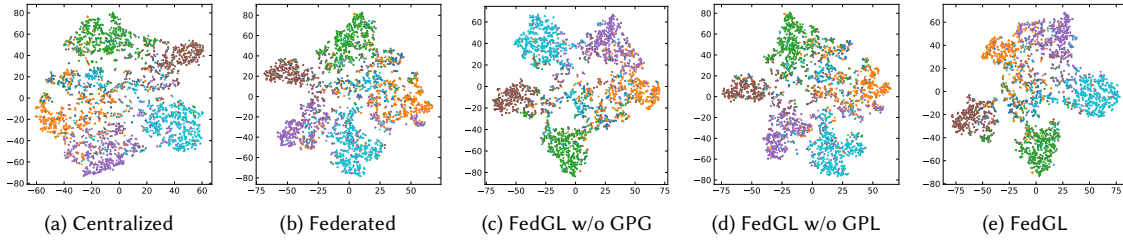
Fig. 6. Visualization of node embedding learned by different methods on Citeseer.

(a) Centralized   (b) Federated   (c) FedGL w/o GPG   (d) FedGL w/o GPL   (e) FedGL
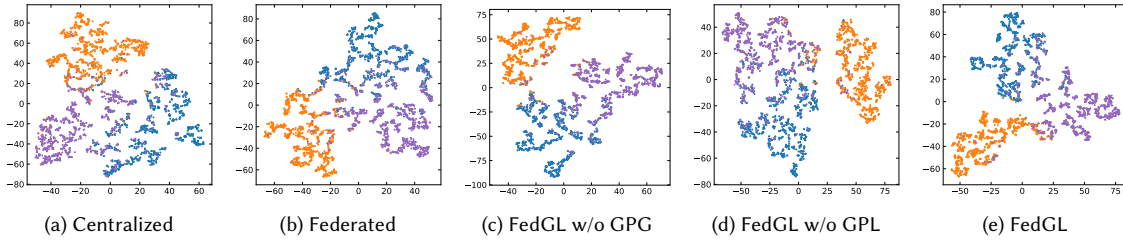
Fig. 7. Visualization of node embedding learned by different methods on ACM.

Cora, Citeseer, and ACM. Since Wiki has 17 classes, it is not easy to display them in color, so Wiki is not reported. In the figures, each scattered point represents a node, and the node with the same color belongs to the same category. As can be seen, the nodes in Centralized are scattered, and there are many overlapping nodes between classes. Federated has fewer overlapping nodes between classes, but the nodes are still scattered and the boundaries between classes are not clear. The nodes in FedGL are quite compact, and the boundaries between classes are clear, which verifies FedGL is capable of learning more discriminative node embeddings, thus performing better than Centralized and Federated in the downstream tasks.

5.5   Q3: Experimental Results under Local Goal

5.5.1   Graph Data Distribution of Clients. In order 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
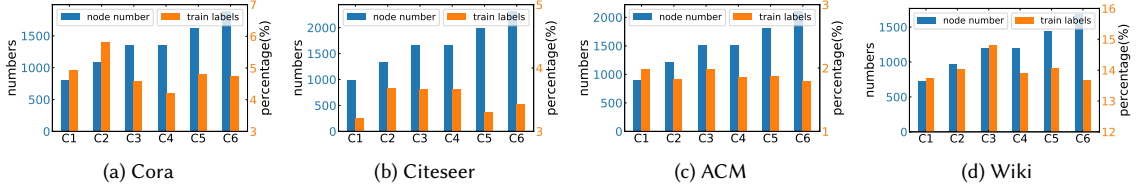
Fig. 8. The number of nodes and the proportion of training labels of each client (C1-C6) on different datasets.
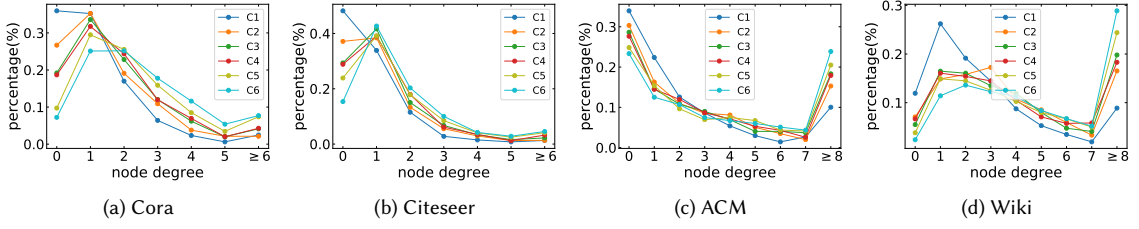

Fig. 9. The node degree distribution of each client (C1-C6) on different datasets.
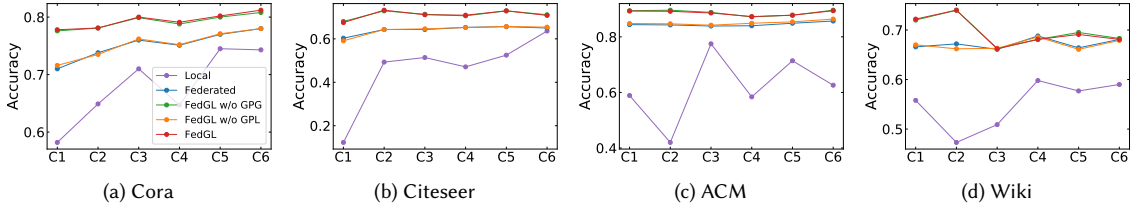

Fig. 10. Node classification accuracy of each client (C1-C6) on different datasets.

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 short, the graph data between clients in the experiment are Non-IID, and there exists highly *heterogeneity* and *complementarity*.

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, we have the following observations.

- Compared with Local, FedGL outperforms it on each client and each dataset, and the performance on each client is not much different and relatively stable, even though clients have 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

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**

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 uses the global pseudo label to enrich the training labels and the global pseudo graph to complement the graph structure, directly improving the quality of each local model and leading to a high-quality 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. It indicates that FedGL can maintain training accuracy while ensuring training speed, and thus can be readily applied to real scenarios. (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%]. Such an observation shows that the data size of clients is more important than the number of clients. Especially when the data of each client are relatively large and there is no magnitude difference in the data size between clients, a higher-quality global model could 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

Table 5. Node classification accuracy under different number of clients and data size.

Dataset	Method	[30%,40%,50%]	[20%,40%,60%]	[50%,60%,70%,80%]	[20%,40%,50%,70%,70%,90%]
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

data between clients and learned a high-quality global model under relatively severe Non-IID situation. (3) Compared to Table 2, FedGL can still achieve promising results and maintains its superiority although clients have the different number of clients and data size.

5.6.3 Overlapping Node Ratio between Clients.
In this experiment, we aim to 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 but control 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.
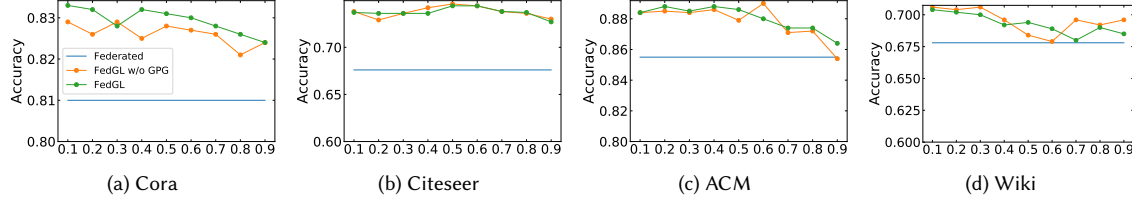
Fig. 11. Node classification accuracy under different confidence threshold λ.
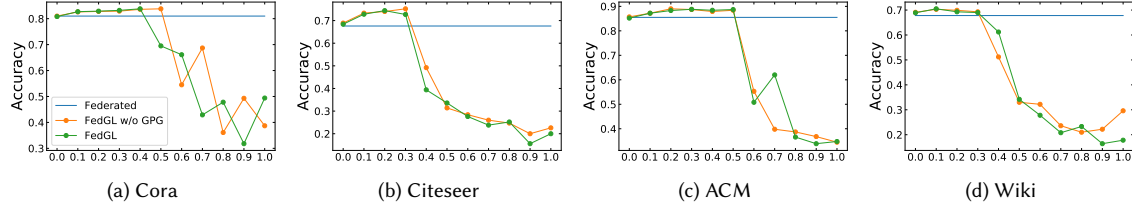
Fig. 12. Node classification accuracy under different self-supervision learning coefficient α.

5.7 Q5: Parameter Sensitivity Analysis

5.7.1 Confidence Threshold. The confidence threshold λ in Eq. (12) is used to control the number of pseudo labels. To explore how λ affects the performance of FedGL, we tune λ in [0.1, 0.9] with step size 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. Especially on Citeseer, the classification accuracy of the global model can be improved by at least 5% as long as the global pseudo label is utilized. For different datasets, FedGL achieves the best performance with different λ, but λ is overall small. From the results, [0.1, 0.3] is a desirable interval.

5.7.2 Self-supervised Learning Coefficient. The self-supervised learning coefficient α in Eq. (8) is used to control the strength of self-supervised learning. To explore the impact of α on FedGL, we tune α in [0, 1] with step size 0.1. Note that $\alpha = 0$ means without 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 the value of 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. When α is relatively small, 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) is used to control the strength of complementing the graph structure. To explore the impact of β on FedGL, we tune β in [0, 1] with step size 0.1. Note that $\beta = 0$ means without using the global pseudo graph, and $\beta = 1$ means that the global pseudo graph is as important as 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.
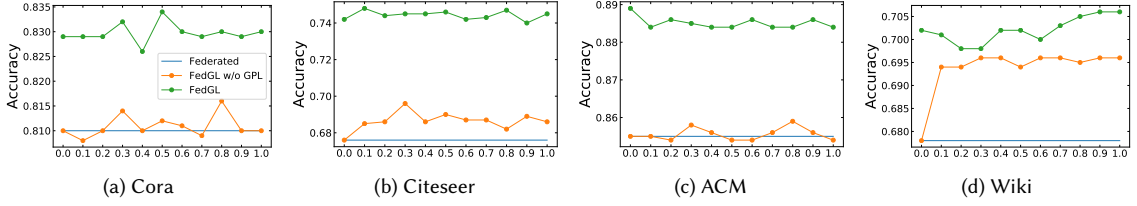
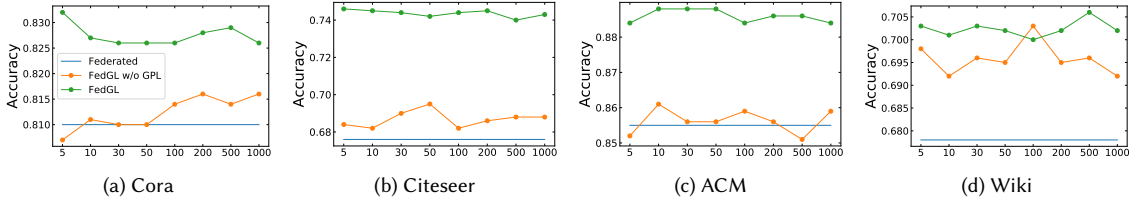Fig. 13. Node classification accuracy under different global pseudo graph coefficient β.

Fig. 14. Node classification accuracy under different neighbor number s of global pseudo graph.

5.7.4 Neighbor Number of Global Pseudo Graph. The neighbor number s is used to control the sparsity of the global pseudo graph. To explore the impact of s on FedGL, we set s to 5, 10, 30, 50, 100, 200, 500, 1000, respectively. The results are shown in Fig. 14. As the value of s 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.

6 CONCLUSION

In this paper, we propose a general federated graph learning framework FedGL, which can collaborate the graph data stored in different clients to train a high-quality graph model while protecting data privacy. To tackle the *heterogeneity* and *complementarity* of graph data between clients, we propose to discover and exploit the global self-supervision information. Concretely, 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 respectively, thereby improving the quality of each local model and obtaining a high-quality global model. More importantly, the process of global self-supervision discovery and using enables the information of each client to flow and share in a privacy-preserving manner, thus mitigating the *heterogeneity* and utilizing the *complementarity*. Finally, extensive experimental results on the node classification task show that FedGL significantly outperforms the centralized method, simple federated method, and local method, which fully verifies the effectiveness of FedGL.

Recall that FedGL is a general federated graph learning framework, which is not limited to specific graph models. Therefore, in the future, we are interested in exploring the effectiveness of FedGL on more graph models such as GAT [47] and FastGCN [5]. Besides, we are also interested in extending FedGL to the scenarios where clients use different graph models and even clients have multimodal data.

REFERENCES

[1] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2938–2948.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[3] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1616–1637.

[4] Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. 2018. Federated meta-learning with fast convergence and efficient communication. *arXiv preprint arXiv:1802.07876* (2018).

[5] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247* (2018).

[6] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. 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*. 257–266.

[7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*. 3844–3852.

[8] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[9] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. 2007. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on knowledge and data engineering* 19, 3 (2007), 355–369.

[10] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1416–1424.

[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[13] Meng Hao, Hongwei Li, Guowen Xu, Sen Liu, and Haomiao Yang. 2019. Towards efficient and privacy-preserving federated deep learning. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[14] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* (2015).

[15] Weibo Hu, Chuan Chen, Yaomin Chang, Zibin Zheng, and Yunfei Du. 2021. Robust graph convolutional networks with directional graph adversarial training. *Applied Intelligence* (2021), 1573–7497. https://doi.org/10.1007/s10489-021-02272-y

[16] Weibo Hu, Chuan Chen, Fanghua Ye, Zibin Zheng, and Yunfei Du. 2021. Learning Deep Discriminative Representations with Pseudo Supervision for Image Clustering. *Information Sciences* 568 (2021), 199–215.

[17] Jie Huang, Chuan Chen, Fanghua Ye, Weibo Hu, and Zibin Zheng. 2020. Nonuniform Hyper-Network Embedding with Dual Mechanism. *ACM Transactions on Information Systems (TOIS)* 38, 3 (2020), 1–18.

[18] Zhihao Jia, Sina Lin, Rex Ying, Jiaxuan You, Jure Leskovec, and Alex Aiken. 2020. Redundancy-Free Computation for Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 997–1005.

[19] Longlong Jing and Yingli Tian. 2020. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[20] Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. 2019. Adaptive gradient-based meta-learning methods. *arXiv preprint arXiv:1906.02717* (2019).

[21] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *NIPS Workshop on Bayesian Deep Learning* (2016).

[22] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[23] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

[24] Dong-Hyun Lee. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, Vol. 3.

[25] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. Deepgcns: Can gcns go as deep as cnns?. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9267–9276.

[26] Yuzheng Li, Chuan Chen, Nan Liu, Huawei Huang, Zibin Zheng, and Qiang Yan. 2020. A blockchain-based decentralized federated learning framework with committee consensus. *IEEE Network* (2020).

[27] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*.

[28] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. 2020. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials* 22, 3 (2020), 2031–2063.

[29] Lumin Liu, Jun Zhang, SH Song, and Khaled B Letaief. 2019. Edge-assisted hierarchical federated learning with non-iid data. *arXiv preprint arXiv:1905.06641* (2019).

[30] Meng Liu, Hongyang Gao, and Shuiwang Ji. 2020. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 338–348.

[31] Wei Liu, Junfeng He, and Shih-Fu Chang. 2010. Large graph construction for scalable semi-supervised learning. In *ICML*.

[32] Yang Liu, Yan Kang, Chaoping Xing, Tianjian Chen, and Qiang Yang. 2020. A secure federated transfer learning framework. *IEEE Intelligent Systems* 35, 4 (2020), 70–82.

[33] Guodong Long, Yue Tan, Jing Jiang, and Chengqi Zhang. 2020. Federated Learning for Open Banking. In *Federated Learning*. Springer, 240–254.

[34] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. PMLR, 1273–1282.

[35] Takayuki Nishio and Ryo Yonetani. 2019. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 1–7.

[36] Caleb C Noble and Diane J Cook. 2003. Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 631–636.

[37] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407* (2018).

[38] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[39] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Process. Mag.* 30, 3 (2013), 83–98.

[40] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. 2017. Federated multi-task learning. *arXiv preprint arXiv:1705.10467* (2017).

[41] Alexander J Smola and Risi Kondor. 2003. Kernels and regularization on graphs. In *Learning theory and kernel machines*. Springer, 144–158.

[42] Michael R Sprague, Amir Jalalirad, Marco Scavuzzo, Catalin Capota, Moritz Neun, Lyman Do, and Michael Kopp. 2018. Asynchronous federated learning for geospatial applications. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 21–28.

[43] Ke Sun, Zhouchen Lin, and Zhanxing Zhu. 2020. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5892–5899.

[44] Zeyi Tao and Qun Li. 2018. esgd: Communication efficient distributed deep learning on the edge. In {*USENIX*} *Workshop on Hot Topics in Edge Computing (HotEdge 18)*.

[45] Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. 2018. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735* (2018).

[46] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[47] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[48] Binghui Wang, Ang Li, Hai Li, and Yiran Chen. 2020. GraphFL: A Federated Learning Framework for Semi-Supervised Node Classification on Graphs. *arXiv preprint arXiv:2012.04187* (2020).

[49] Hao Wu and Saurabh Prasad. 2017. Semi-supervised deep learning using pseudo labels for hyperspectral image classification. *IEEE Transactions on Image Processing* 27, 3 (2017), 1259–1270.

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

[51] Jie Xu, Benjamin S Glicksberg, Chang Su, Peter Walker, Jiang Bian, and Fei Wang. 2021. Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research* 5, 1 (2021), 1–19.

[52] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, Vol. 80. 5449–5458.

[53] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.

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

[55] Naoya Yoshida, Takayuki Nishio, Masahiro Morikura, Koji Yamamoto, and Ryo Yonetani. 2020. Hybrid-FL for wireless networks: Cooperative learning mechanism using non-IID data. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 1–7.

[56] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. 2020. When does self-supervision help graph convolutional networks?. In *International Conference on Machine Learning*. PMLR, 10871–10880.

[57] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. 2021. A survey on federated learning. *Knowledge-Based Systems* 216 (2021), 106775.

[58] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2018. Network representation learning: A survey. *IEEE transactions on Big Data* 6, 1 (2018), 3–28.

[59] Jun Zhou, Chaochao Chen, Longfei Zheng, Xiaolin Zheng, Bingzhe Wu, Ziqi Liu, and Li Wang. 2020. Privacy-preserving graph neural network for node classification. *arXiv preprint arXiv:2005.11903* (2020).

[60] Hankz Hankui Zhuo, Wenfeng Feng, Qian Xu, Qiang Yang, and Yufeng Lin. 2019. Federated reinforcement learning. *arXiv preprint arXiv:1901.08277* 1 (2019).

[61] Daniel Zügner and Stephan Günnemann. 2020. Certifiable robustness of graph convolutional networks under structure perturbations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1656–1665.