Neural Networks 169 (2024) 143–153

Contents lists available at ScienceDirect

Neural Networks

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

BI-FedGNN: Federated graph neural networks framework based on Bayesian inference

Rufei Gao [a], Zhaowei Liu [a,*], Chenxi Jiang [a], Yingjie Wang [a], Shenqiang Wang [b], Pengda Wang [c]

[a] School of Computer Science and Engineering, Yantai University, Shandong, China
[b] Institute of Network Technology (Yantai), Shandong, China
[c] University of Science and Technology of China, Anhui, China

ARTICLE INFO

Keywords:
Graph neural networks
Industrial Internet of Things
Federated graph learning
Bayesian inference

ABSTRACT

The development of the Industrial Internet of Things (IIoT) in recent years has resulted in an increase in the amount of data generated by connected devices, creating new opportunities to enhance the quality of service for machine learning in the IIoT through data sharing. Graph neural networks (GNNs) are the most popular technique in machine learning at the moment because they can learn extremely precise node representations from graph-structured data. Due to privacy issues and legal restrictions of clients in industrial IoT, it is not permissible to directly concentrate vast real-world graph-structured datasets for training on GNNs. To resolve the aforementioned difficulties, this paper proposes a federal graph learning framework based on Bayesian inference (BI-FedGNN) that performs effectively in the presence of noisy graph structure information or missing strong relational edges. BI-FedGNN extends Bayesian Inference (BI) to the process of Federal Graph Learning (FGL), adding random samples with weights and biases to the client-side local model training process, improving the accuracy and generalization ability of FGL in the training process by rendering the graph structure data involved in GNNs training more similar to the graph structure data existing in the real world. Through extensive experimental tests, the results show that BI-FedGNN has about 0.5%–5.0% accuracy improvement over other baselines of federal graph learning. In order to expand the applicability of BI-FedGNN, experiments are carried out on heterogeneous graph datasets, and the results indicate that BI-FedGNN can also have at least 1.4% improvement in classification accuracy.

1. Introduction

GNNs have been extensively utilized to model graphically structured data in a variety of scenarios and applications, including recommendation systems (Sun et al., 2023; Wu, Sun, Zhang, Xie, & Cui, 2022), node classification (Shen, Pan, Choi, & Zhou, 2023; Wang et al., 2023), and drug discovery (Gaudelet et al., 2021). With the continuous increase of graph-structured data, how to retain rich graph topology information while ensuring a small computational cost is an urgent problem, and graph-based representation learning methods can help solve this problem (Tang et al., 2022, 2023). Most existing GNNs follow the principle of centralized training, which requires the collection of graph data before training (Li, Liu, Ma, Yang, & Sun, 2022; Xu, Hu, Leskovec, & Jegelka, 2019) to select and learn node features (Yan et al., 2023; Zhou, Song, Yu, & Zheng, 2023). However, the vast majority of graph structure information in IIoT environments contains private information about the user, and this information is held by different information managers and cannot be trained directly, preventing traditional GNNs from training powerful models through collective intelligence (Chang et al., 2023; Zhang et al., 2021). In the meantime, graph structure information in the IIoT is typically extracted from complex interaction systems that contain uncertainties or errors, resulting in the prevalence of missing, meaningless, or even spurious edges in the graph structure data, which affects the precision or accuracy of GNN training. As shown in Fig. 1. How to effectively solve distributed, error-containing graph structure data has a significant impact on machine learning in an IIoT environment.

As a new distributed machine learning paradigm, Federated Learning (FL) allows clients to collaboratively train globally shared models or personalized models in a decentralized manner through various privacy-preserving strategies (Wang, Wang, et al., 2022) without contributing their own local data (Tun, Nguyen, Thwal, Choi, & Hong, 2023). This feature enables the application of FL to graph data to alleviate the data isolation problem and secure the proprietorship of

* Corresponding author.
E-mail addresses: gaorufei@s.ytu.edu.cn (R. Gao), lzw@ytu.edu.cn (Z. Liu), 17863565053@s.ytu.edu.cn (C. Jiang), wangyingjie@ytu.edu.cn (Y. Wang), wsqaahh@foxmail.com (S. Wang), wangpengda@cetccloud.com (P. Wang).

https://doi.org/10.1016/j.neunet.2023.10.024
Received 19 June 2023; Received in revised form 1 October 2023; Accepted 17 October 2023
Available online 18 October 2023

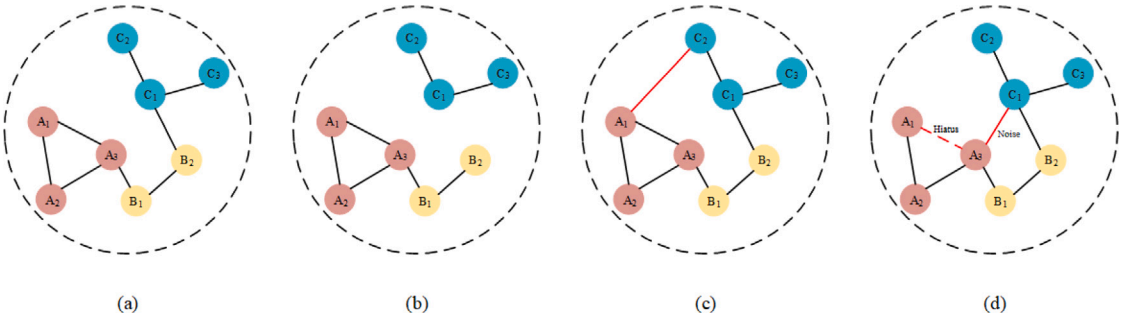


Fig. 1. The differences between real-world graph structure data and actual trained graph structure data include noise and the absence of strong relationship edges. (a) represents graph structure data in the real world; (b), (c), and (d) are three types of biased data, respectively. (b) represents the missing problem of strong relational edges, i.e., B_2 and C_1actually have one edge, but the edge may be missing in the extraction process in the complex industrial internet of things environment; (c) is a problem with noise edges, i.e., A_1 and C_2 In practice, there are no edges, but due to the influence of noise, the extracted training data has a noisy edge; (d) is a mixture of two problems, namely A_1 and A_3 The actual edge is missing, A_3 and C_1 There is actually no edge, but there is a noisy edge.

graph data for each client. Federated graph learning, an application of FL to graph data (Fu, Zhang, Dong, Chen, & Li, 2022), has emerged as a highly promising avenue for further investigation of GNNs on decentralized graph data. The benefit of FGL is that it enables users to maintain their own private data locally while enhancing overall performance through collaboratively learning globally shared models (He, Ceyani, Balasubramanian, Annavaram, & Avestimehr, 2022; Wu, Wu, et al., 2022). This mechanism enables FGL to address the privacy and security concerns associated with graphical data in order to avoid the issues and risks associated with data centralization, which is essential for information processing in the IIoT. Therefore, FGL has a prospective application and can play a significant role in social networks, recommender systems, industrial IoT, etc. Additionally, it provides researchers with a new research direction.

The Non-iid data issue is a significant obstacle in FL (Jing et al., 2023; Shu et al., 2023). Depending on the client's behaviors or preferences, the client's local dataset may be vastly different, which might result in training instability and accuracy degradation. For instance, client data may originate from a variety of domains (Kairouz et al., 2021; Li, JIANG, Zhang, Kamp, & Dou, 2021), resulting in vastly different distributions of datasets. Due to the diversity of graph data, unfortunately, the issue of Non-iid data also exists in FGL (Xie, Ma, Xiong, & Yang, 2021). Different companies or platforms, for instance, maintain vast amounts of social network data, which vary significantly due to diverse data acquisition methods and purposes. In the interim, graph data in GNNs may contain issues such as absent and meaningless information, a challenge also faced by FGL (Fang & Ye, 2022; Peng, Wang, Dvornek, Zhu, & Li, 2022). A large quantity of data noise or missing graph data may lead to errors in GNN's model training based on their own local subgraphs by users participating in FGL, which in turn leads to inaccurate results for FGL as a whole. Consequently, overcoming Non-iid data and optimizing the graph structure are extremely advantageous for FGL.

Nevertheless, learning the most suitable graph structure for GNNs is a difficult endeavor that necessitates the resolution of two obstacles. The first problem is how to generate the graph. Most of the current approaches parameterize each edge locally (Franceschi, Niepert, Pontil, & He, 2019; Jiang, Zhang, Lin, Tang, & Luo, 2019) without considering the underlying generation mechanism of the graph. As a result, these methods are less tolerant of noise and sparsity. Solving this problem requires considering the graph generation mechanism (Xie & Tu, 2022), for example, using methods such as configuration models (Casiraghi & Nanumyan, 2021) to drive the global structure of the learned graph while keeping the rules, thus making the learned graph more robust. The second issue is how to inject multiparty information. Learning graph structure from a single data source can lead to inaccurate graphs. To improve the reliability of the graph, more comprehensive information should be taken into account. Current methods (Jin et al., 2020) mainly use feature similarity to inject multiparty information, but this approach has some limitations. A superior method should incorporate data from multiple data sources to increase the precision and robustness of the learned graphs. In summary, solving the above two problems is the key to learning the optimal graph structure of GNNs. Future research can explore more effective graph generation methods and strategies for injecting multi-party information to further improve the performance of GNNs.

To further address the training effect of data differences on the overall model during federal graph learning to improve the accuracy of node classification. In this paper, Bayesian inference-based federal graph learning (BI-FedGNN) is proposed. Unlike the traditional federal graph learning process, BI-FedGNN reduces the impact of noise or missing edges on model training by using Bayesian inference in the process of GNN model training based on the client's own local subgraph. During the training process, Gaussian-distributed weights and biases are introduced to the node features that make up the model, and these randomly sampled weights and biases are distributed a priori and a posteriori by BI to obtain different model instances, which makes the trained model possess uncertainty capability. To prevent training bias, information shared by other clients participating in FL is injected as multi-order neighborhood information into the model training procedure. The above multi-view information is jointly considered as the best graph information of GNNs for client-side local training. The training parameters are subsequently uploaded to the central server, which aggregates them and transmits them to clients who are participating in the training sessions so that their models can be updated. This procedure is repeated until the model converges, thereby increasing the precision of FGL training as a whole. The following are the three principal contributions of this paper:

- This paper applies the BI method to the process of local model training for FGL clients, enabling the clients participating in FGL training to obtain better GNN models when training based on local subgraphs, thereby improving the overall training efficiency of the federal graph learning framework.
- BI-FedGNN, a novel federal graph learning framework, is proposed in this paper, which improves the uncertainty of local model training by adding randomly sampled weights and biases to the subgraphs owned by the client. In addition, it combines multi-order neighborhood information with information shared by other clients during the client's local model training, increasing the accuracy of the model's training and reducing the impact of data noise or missing strong relational edges on the training of the FGL model.
- In this paper, a regular term for KL divergence loss is added to the loss function to prevent the model from being overfitted during

training, and the efficacy of BI-FedGNN is validated by comparing it to the state-of-the-art federal graph learning framework in a number of challenging benchmark tests.

2. Related work

On the basis of the work presented in this paper, a concise summary of the most relevant work on Graph Neural Networks, Federal Graph Learning, and Bayesian Inference is provided.

2.1. Graph neural networks

Graph Neural Networks are an effective neural architecture for mining graph structured data, extracting and uncovering features and patterns in graph structured data, while capturing higher order content and topological information on the graph. Most of GNNs discover their representation of nodes through aggregating and transforming the embeddings of adjacent nodes and themselves. For instance, GCN (Wan, Yuan, Zhan, & Chen, 2022) aggregates messages by averaging over neighboring representations, GAT (Ding et al., 2023) aggregates characteristics of neighboring nodes to the central vertex based on attention coefficients, and GIN (Xu et al., 2019) aggregates data using summation functions. GNNs can achieve outstanding performance when learning a set of graphs in a shared feature space when employing such message-passing features. However, the feature-based message initialization mechanism tightly couples the feature space and learnable parameters in GNNs, making it difficult to train GNNs on graph data from multiple domains (Liu, Ding, Liu, & Pan, 2023; Qiu et al., 2020).

2.2. Federated graph learning

Federated Learning has garnered significant attention for its potential to facilitate collaborative training while protecting data privacy (Kairouz et al., 2021; Liang et al., 2023). Traditional federation learning algorithms, such as FedAvg (McMahan, Moore, Ramage, Hampson, & y Arcas, 2017), perform local training on the client side, global parameter averaging on a central server, and so on iteratively until the model converges. However, the current assumptions of federation learning are based on some completely "clean" datasets and do not consider the problem of label noise in the data (Fang & Ye, 2022). Due to the different behaviors and habits of customers in the real world, each customer's data will have different labels and features, resulting in data heterogeneity (Wang, Liu, Xu, & Yan, 2022), which presents another significant challenge in FL: federated learning with non-iid data (Li et al., 2020). Federated Graph Learning combines FL with graph learning to support distributed GNN training and extend the application scenario of GNNs (Fu et al., 2022). Existing FGL can be classified into three types:inter-graph FGL (Xie et al., 2021), intra-graph FGL (Pei et al., 2021), and graph structure FGL (Meng, Rambhatla, & Liu, 2021). In inter-graph FGL, each client possesses a collection of graphs and participates in federated training to acquire more effective GNNs for modeling local models. The most common use of the inter-graph FGL framework is within the biochemistry industry, in which data sharing between companies is not possible due to commercial competition but is possible under the inter-graph FGL framework. In intra-graph FGL, every client possesses a portion of the graph's data (i.e., sub-graphs), which can be specifically divided into horizontal FGL (Tan et al., 2022) and vertical FGL (Chen, Zhou, et al., 2022). The subgraph data held in each client in the horizontal FGL can be seen as divided horizontally from the underlying whole graph (the connections between the subgraphs are lost and can strictly overlap). Each subgraph in the vertical FGL is parallel, has a large overlap with each other, and can be seen as being divided from the underlying whole graph divided vertically. Intra-graph FGL can be applied to financial crime detection. In graph structured FGL, graphs are used to model the topological relationships between clients, i.e., to aggregate local models using GNNs based on the topological relationships of clients. Examples of common application circumstances include images (Chen, Long, et al., 2022) and traffic data. This paper focuses primarily on intra-graph FGL, with an emphasis on learning better local models to enhance the overall computational precision of FGL.

2.3. Bayesian inference

Bayesian Inference is an approach to statistical inference based on Bayes' theorem, which can infer the probability distribution of an unknown quantity based on known data and prior knowledge. In Bayesian inference, the prior probability distribution represents the initial beliefs about the unknown quantity, while the posterior probability distribution represents the update of beliefs about the unknown quantity after the data are observed. By continuously iterating this process, the uncertainty can be gradually reduced, and increasingly accurate probability distributions can be obtained. The combination of Bayesian inference and graph convolutional network (GCN) can be used to infer the potential representation of nodes and edges in graph data. Traditional GCN methods typically utilize the information of local neighbor nodes to update the representation of each node; however, this approach may disregard the global topology and the information of interfering nodes. Incorporating BI into the model training procedure of GNNs (Liu, Yang, Wang, Lu, & Li, 2023; Liu, Yang, Wang, & Su, 2022; Wang et al., 2021) can not only take into account the global topology, but also prevent overfitting and enhance the model's robustness by utilizing the prior distribution. Specifically, these methods generally treat node representations as random variables and use BI to update the posterior probability distribution of node representations. Through continuous iterative updates, more robust and accurate node representations can be obtained. This combination of methods has a wide range of applications in industrial IoT, social networks, recommender systems, bioinformatics, and other fields.

3. Methodology

In this chapter, BI-FedGNN, a framework for federated graph learning based on Bayesian inference, is introduced in detail.

3.1. Research objectives

The implementation background of this article is that in the industrial Internet of Things environment, most of the data in the Industrial Internet of Things is owned by different data holders, and centralized training is not feasible. The goal is to improve the uncertainty of the client's local model and use Bayesian inference to eliminate the impact of client local data participating in training on local model training due to noise or uncertain information. At the same time, KL-Loss is used as the regularization term of the overall loss function to prevent over-fitting problems in training. Thereby improving the training effect of federated graph learning in the industrial Internet of Things environment.

3.2. Problem formulation

Let $G = (V, E, X)$ be a graph, where V represents a set of nodes, $V = \{v_1, v_2, \ldots, v_n\}$, E is the group of edges, $X = \{x_1, x_2, \ldots, x_n\}$ represents the characteristic matrix of the node, x_i is the characteristic vector of node v_i. Relationships between nodes, which can be represented by a critical matrix A, are characterized by edges, where A_{ij} represents the relationship between nodes v_i and v_j.$Y_n = \{y_1, y_2, \ldots, y_n\}$ is the label of the node, where y_i is the label of node v_i. Through graph G and label Y_N, GNNs can learn the optimal adjacency matrix $S \in S = [0, 1]^{N \times N}$ and GNNs parameters Θ, to enhance the performance of node

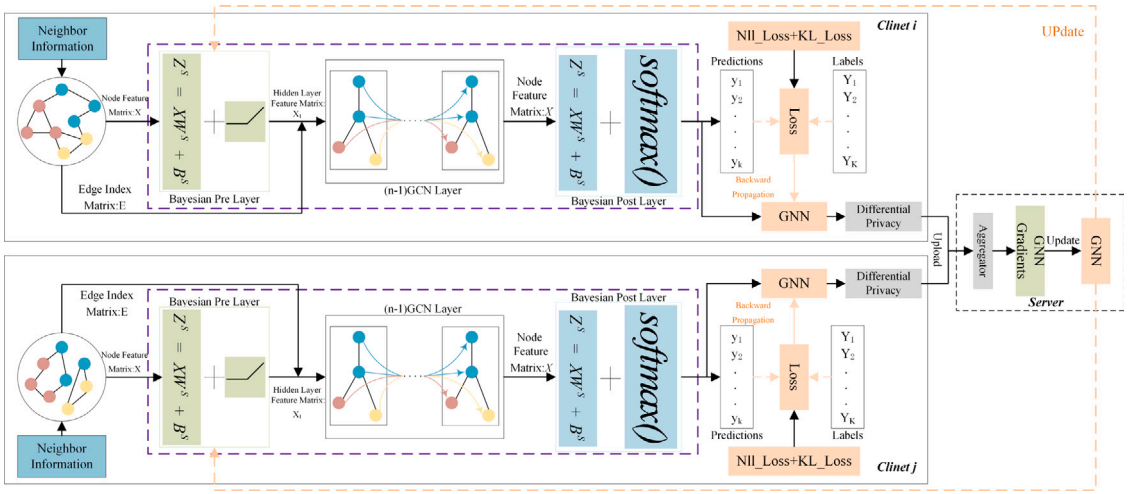


Fig. 2. The basic structure of the BI-FedGNN model. BI-FedGNN is divided into two parts, namely client local model parameter training and server global model parameter aggregation. The client uses a Bayesian GNN model to improve the uncertainty of local model training by randomly sampling weights and biases that conform to a Gaussian distribution during local model training. At the same time, in order to avoid the problem of over-fitting, the KL divergence loss is added as a regularization term to the overall loss function. The differential privacy method is used to upload local parameters to the server, and the server uses the Fedavg strategy to achieve aggregation of global parameters.

classification for unmarked nodes. The function of objectiveness can be expressed as follows:

$$min\mathcal{L}(A, X, Y_N) = \sum_{v_i \in V} \Gamma(f\Theta(X, S)_i, yi) \tag{1}$$

Where $f\Theta(X, S)_i$ is the prediction of node v_i, and $\Gamma(pre, lable)$ is the difference between the measured value and the true label, such as KL divergence loss.

3.3. Overview

During training, the majority of FGL approaches consider the graph owned by each client to be the most accurate description of the relationships between nodes. However, data in industrial IoT scenarios is frequently imperfect, with some false edges or absent strong relational edges in the graph, causing FGL to produce subpar classification results. In this paper, a Bayesian inference-based approach is proposed to mitigate the impact of noise or missing edges in the training data on FGL. This model can model the uncertainty of weights and biases, with prior and posterior distributions, to enhance the model's reliability and precision. An example of the BI-FedGNN model is shown in Fig. 2. There are a variety of FGL models currently available, but this paper focuses on the GCN-based federated graph learning framework.

3.4. Bayesian linear layer

Each parameter is modeled as a variable at random with prior and posterior distributions. In addition to optimizing the posterior distribution of the parameters during training, the posterior distribution of the parameters is also estimated as the output of the model. Thus, during inference, it is possible to sample from the posterior distribution of the parameters to acquire multiple instances of the model, thereby capturing the model's uncertainty. This method can better consider the model's generalization ability on unseen data, thereby improving the model's robustness and reliability.

In the Bayesian linear layer, the characteristic matrix of the graph is first initialized with weights $W \in \mathbb{R}^{D_{in} \times D_{out}}$ and biases $B \in \mathbb{R}^{D_{out}}$. These weights and biases are initialized using a random Gaussian distribution, and linear transformations of input data are achieved using different weights and biases to obtain better model output. At the same time, the weights and biases in this layer are assigned a priori and a posteriori probability distribution. Assume that the input of the Bayesian linear layer is $X \in \mathbb{R}^{n \times d}$, where n represents the total amount of nodes and d represents the dimension of each node's feature vector. The weights and biases of this layer obey a Gaussian distribution with mean 0 and standard deviations of σ_w and σ_b, respectively. The specific forward propagation process can be expressed as:

$$Z = XW + B \tag{2}$$

Wherein $W \in \mathbb{R}^{d \times h}$ represents the weight matrix, $B \in \mathbb{R}^{h}$ represents the offset vector, and $Z \in \mathbb{R}^{n \times h}$ represents the output of the layer. However, in the Bayesian linear layer, both W and B are random variables, so it is necessary to give their prior distribution $p(\theta)$. Taking $p(W)$ as an example, $p(B)$ is also similar. In this paper, the Gaussian distribution is used as a prior distribution:

$$p(W|\alpha) = \mathcal{N}(0, \alpha^{-1} I) \tag{3}$$

Where α is a hyperparameter that controls the variance of the prior distribution, and I is the identity matrix. For each input X, a posterior distribution can be obtained by substituting a prior distribution into Bayesian inference:

$$\begin{aligned} p(W|X, Y) &= \frac{p(Y|X, W)p(W|\alpha)}{p(Y|X)} \\ &= \mathcal{N}((\mu = \frac{1}{\sigma^2} \cdot \Sigma \cdot X^T Y), \Sigma = (\frac{1}{\alpha} \cdot I + \frac{1}{\sigma^2} \cdot X^T X)^{-1}) \\ &= \mathcal{N}(\mu, \Sigma) \end{aligned} \tag{4}$$

Therefore, a posterior distribution is also a Gaussian distribution, where $Y \in \mathbb{R}^{n \times c}$ represents the label of the sample, c represents the number of categories, and μ and Σ are the mean and covariance matrices of the posterior distribution, respectively. By randomly sampling a posterior distribution, a set of weight parameters W^s can be obtained. By substituting W^s into the forward propagation formula, the corresponding output Z^s can be obtained:

$$Z^S = XW^S + B^S \tag{5}$$

Where B^s represents the offset vector sampled from a prior distribution. Using the Monte Carlo technique, the final output of this layer can be obtained by averaging the sampled multiple sets of output Z^s:

$$Z = \frac{1}{S} \sum_{s=1}^{S} Z^S \tag{6}$$

S represents the number of total samples, and finally transmits the output Z of this layer to the next layer for further forward propagation. At the same time, this work also randomly samples a set of

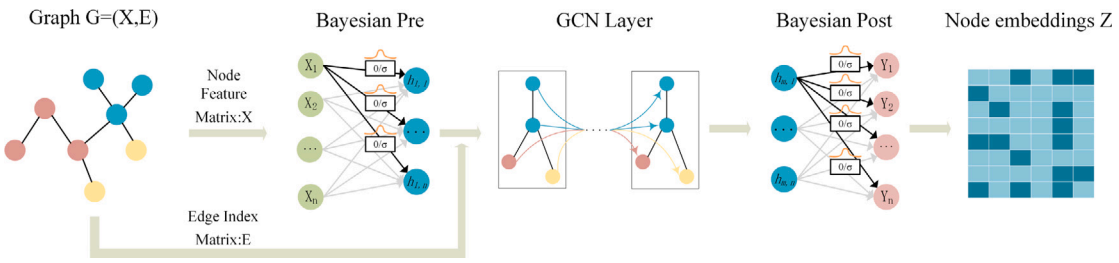


Fig. 3. Overall framework of Bayesian GNN module.

weighted and biased noise to add to the final weights and biases, and re-parameterize the weights and biases. At this time, the weights and biases are random variables, not determined values. During each forward propagation process, each calculation will sample from the distribution of the weights and biases, so the results obtained from each forward propagation are also random. The purpose is to introduce randomness to represent the uncertainty of weights and biases, so as to better and more accurately describe the uncertainty of the model. Specifically, it can be expressed as:

$$w = \mu_w + \sigma_w \odot \epsilon_w \tag{7}$$

$$b = \mu_b + \sigma_b \odot \epsilon_b \tag{8}$$

Where μ_w, μ_b is the mean value of weight and bias, σ_w, σ_b is the standard deviation of weight and bias, ϵ_w, ϵ_b is the noise value of random sampling weight and bias, and $\odot$ is the multiplication of elements.

3.5. Bayesian GNN module

There are many existing GNNs models, such as GAT, GCN, GraphSAGE, and so on. This article selects a representative GCN as the backbone for local client model training in federated graph learning. When the client performs training based on local subgraphs, the original graph $G = (V, E, X)$ is input into the Bayesian GCN module for local node classification training. After training is complete, the model parameters are transmitted to the central server using differential privacy. The central server aggregates and then issues all parameters. The client then completes the local model update based on the newly issued parameters. This process is iterative until the final model converges. The detailed model is shown in Fig. 3.

Specifically, GCN employs the neighborhood aggregation strategy, which repeatedly updates the representation of a node by aggregating its neighborhood representation and its own data. The L-layer aggregation rule of GCN can be expressed as:

$$H^{(L)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(L-1)}W^{(L)}) \tag{9}$$

Wherein, $\tilde{A} = A + I_N$ is the normalization matrix of adjacency of graph G that has an added self-relationship, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. W is a trainable weight matrix, σ is an activation function, and GCN model parameter $\Theta = (W^{(1)}, W^{(2)}, \ldots, W^{(L)})$ can be trained by gradient descent. The matrix represented by nodes in layer L^{th} is $H^{(L)} \in \mathbb{R}^{N\times d}$, and in layer 0 there is $H^{(0)} = X$. The activation function of the final layer of an L-layer GCN is softmax, and the ultimate prediction result is:

$$Z = f(X, A) = softmax(\tilde{A}\,\mathrm{ReLU}(\tilde{A}XW^{(0)})W^{(L-1)}) \tag{10}$$

In this paper, the influence of subgraph data owned by the local client containing some noise or the lack of strong relational edges on model training is considered. Instead of using the traditional deterministic linear layer to process the data, the Bayesian linear layer proposed in Section 4.3 is used, which can provide uncertainty estimates of the model and can also effectively prevent overfitting problems to a certain extent, thus improving the overall robustness and generalization ability of the FGL framework.

Specifically, input the node feature matrix X and edge index matrix E of graph G into the Bayesian GCN model. Firstly, input the node feature matrix X into the Bayesian Pre layer, where weights and biases are randomly sampled and combined with the incoming node feature matrix to map node moments to the hidden layer feature matrix X_{nhid} in the $nhid$ dimension. Subsequently, the hidden layer feature matrix X_{nhid} and edge index matrix E are input into the $(n-1)$ layer GCNConv layer, and each layer GCNConv layer performs convolution operations on the input data, ultimately outputting an $nhid$ dimensional node feature matrix X_{NFM}. Input X_{NFM} into the Bayesian Post layer, and the node feature matrix will be mapped to the output feature matrix X_{nclass} of the nclass dimension. By employing the softmax function, it is possible to determine the probability distribution of each node belonging to each category. By using the backpropagation function, the gradient of trainable parameters in the model trained by the client based on local subgraphs can be obtained. The gradient is then uploaded to the centralized server in order to aggregate global parameters for the model. The client is then sent the global model parameters for the following round of training, and this procedure is repeated until all of the models converge. As this article mainly focuses on intra-graph FGL, which is a horizontal federated graph learning model, it can be represented as:

$$\mathrm{w}_{t+1} = \sum_{m=1}^{M} \frac{N_m}{N} \cdot \mathrm{w}_{t,m} \tag{11}$$

Among them, $\mathrm{w}_{t,m}$ is the local model parameter of the m client after t rounds of updates, N_m represents the amount of data transmitted by client m, N is the sum of the numbers of all clients, and w_{t+1} is the global model parameter aggregated by the central server.

3.6. Regularization of loss function

In this paper, the model's loss function consists primarily of two components: negative logarithmic likelihood loss and KL divergence loss. The negative logarithmic likelihood loss measures the difference between the predicted results of the model and the actual label. This loss converts the difference between the probability distribution of the model prediction and the one-hot encoding of the actual label into a scalar value, which can be shown as follows:

$$\mathcal{L}_{nll} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij}) \tag{12}$$

Among them, $\hat{y}$ represents the output of the model, y represents the real label, N represents the quantity of specimens, C indicates the number of classes, y_{ij} represents whether the i sample belongs to category j, and $\hat{y}_{ij}$ represents the prediction probability of the model for the i sample belonging to category j. The KL divergence loss quantifies the difference between the model's predicted and prior distributions. KL divergence is a metric for measuring the distance between two

Table 1
Synopsis of datasets.

Datasets	Nodes	Edges	Classes	Features	Train	Val	Test
CiteSeer	3327	12,431	6	3703	100	100	800
PubMed	19,717	108,365	3	500	100	100	800
Coauthor-CS	18,333	182,121	15	6805	100	100	800
Coauthor-Physics	34,493	530,417	5	8415	100	100	800
DBLP	26,128	239,566	4	4231	100	100	800

probability distributions, which limits the parameter space of a model. In particular, it can be stated as:

$$\mathcal{L}_{kl} = KL(q(\theta_i) \parallel p(\theta)) = \frac{1}{2}\sum_{j=1}^{D}(\log\frac{\sigma_j^2}{\tilde{\sigma}_j^2} + \frac{\tilde{\sigma}_j^2 + (\mu_j - \tilde{\mu}_j)^2}{\sigma_j^2}) \tag{13}$$

Among them, $q(\theta_i)$ represents the posterior distribution of the i sample, $p(\theta)$ represents the prior distribution, μ_j and σ_j represent the mean and standard deviation of the posterior distribution, $\tilde{\mu}_j$ and $\tilde{\sigma}_j$ represent the mean and standard deviation of the prior distribution, and D indicates the total amount of model parameters. Therefore, the total loss function can be expressed as:

$$\mathcal{L} = \mathcal{L}_{nll} + \mathcal{L}_{kl} \tag{14}$$

As a regularization term, the KL divergence loss is incorporated into the total loss function. By controlling the size of the regularization coefficient, the complexity of the model can be controlled, and then the fitting ability and generalization ability of the model can be controlled.

In general, NLL-loss measures the difference between the real data labels and the labels predicted by the Bayesian GNN model to prevent the labels predicted by the model from being far different from the actual labels. At the same time, since the weights and biases added to nodes in this article are randomly sampled, KL-loss is used to measure the difference between the predicted distribution and the prior distribution of the model to prevent over-fitting problems. Therefore, KL-loss is used as a regularization term of the overall loss function, and NLL-loss and KL-loss are combined for better training of the model.

4. Experiment

In this section, a significant quantity of experiments will be conducted to evaluate the effectiveness of BI-FedGNN. In the FGL node classification task in the graph, BI-FedGNN was compared with existing methods to evaluate whether the classification results of BI-FedGNN were more accurate, and the mechanism and properties of the proposed model were further analyzed.

4.1. Experimental setup

In this subsection, the datasets, baseline, and experimental specifics utilized in the experiment are described.

4.1.1. Datasets

This work validated the proposed BI-FedGNN on five open graph datasets. Firstly, the dataset was processed to generate subgraphs centered around a certain node for model training in FGL. Classify the graph data structure based on whether or not it is isomorphic. Table 1 provides a summary of the dataset's statistical information.

- CiteSeer (Giles, Bollacker, & Lawrence, 1998) and PubMed (Sen et al., 2008) are literature reference network datasets. Nodes represent the literature of the paper, while edges represent the citation relationship. Labels are academic disciplines, and node characteristics are a bag-of-words depicting documents.
- Coauthor-CS (Shchur, Mumme, Bojchevski, & Günnemann, 2018) and Coauthor-Physics (Shchur et al., 2018) are datasets that contain coauthor ship relationships. Authors are represented by nodes, while coauthor ship relationships are represented by edges, labels represent the number of paper categories, and node characteristics are a bag-of-words representation of papers' essential phrases.
- DBLP (Tang et al., 2008) is a collection of authors retrieved from the DBLP website. If it is a coauthor relationship, there is an edge between the two authors.

Please note that based on isomorphism, these data are divided into two categories, with CiteSeer, PubMed, Coauthor CS, and Coauthor Physics being isomorphic datasets and DBLP being heterogeneous datasets.

4.1.2. Baselines

In order to better evaluate BI-FedGNN, it was compared with four baseline federated graph learning methods, including GCN-FGL, GAT-FGL, SAGE-FGL, and SGC-FGL (He et al., 2021). The final comparison results were obtained through the Micro-F1 and Macro-F1 values.

- GCN-FGL: During local model training, the feature vectors of the nodes are derived by considering the attributes of the nodes themselves and the attributes of the nodes' neighboring nodes.
- GAT-FGL: FGL framework for applying attention operations to graph nodes during client-side local model training.
- SAGE-FGL: The framework is comprised of sampling and aggregation, first sampling the neighbors using the connection information between nodes and then continuously fusing the information of neighboring nodes through a multi-layer aggregation function.
- SGC-FGL: A simplified version of the GCN that eliminates nonlinearity and collapses the weight matrix to reduce the additional complexity of the original GCN.

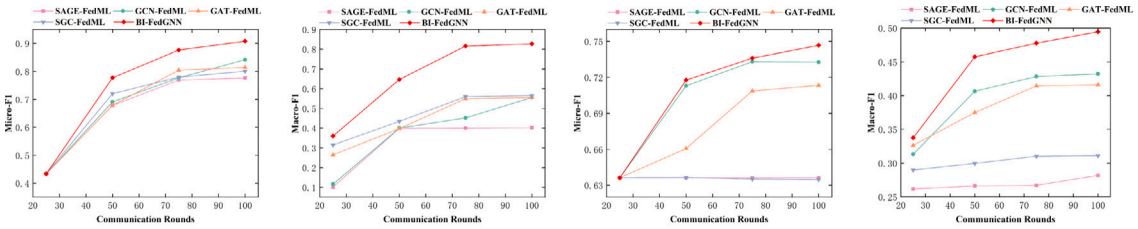
4.1.3. Implementation details

All the experiments were run on a GPU server with two NVIDIA GeForce RTX 3090 GPUs and a 12th Gen Intel(R) Core(TM) i9-12900K 24-core processor. Python and PyTorch have respective versions of 3.8.0 and 1.11.0.

All models have a hidden size of 32, node-embedding-dim is set to 32, graph-embedding-dim is set to 64, readout-hidden-dim is set to 64, and the training level of the model is 5. Using the SGD optimizer with a weight decay of 1e-4, the learning rate is 0.03. At the same time, the ReLU function is used as the function to activate and the dropout rate is set to 0.3 to prevent overfitting even further. Through test comparison, when the number of communication rounds is less than 100, the data quality changes greatly. When it is greater than 100, the data quality changes very little, so the number of communication rounds for all FGL methods is 100. The data protection strategy adopted in this article when the client uploads local training parameters is the traditional differential privacy method.

Table 2
Node classification results (%). (Index: Micro-F1 Macro-F1 Bold: best.)

Datasets	Metrics	GCN-FedML	SAGE-FedML	SGC-FedML	GAT-FedML	BI-FedGNN
CiteSeer	Micro-F1	84.22	77.65	80.10	81.64	**90.79**
	Macro-F1	55.51	40.17	56.53	55.52	**82.69**
PubMed	Micro-F1	84.3	84.02	81.32	82.05	**84.49**
	Macro-F1	76.52	77.87	74.63	78.35	**79.60**
Coauthor-CS	Micro-F1	74.39	65.02	74.83	71.15	**75.99**
	Macro-F1	41.19	32.74	**47.68**	45.22	47.59
Coauthor-Physics	Micro-F1	82.62	72.71	87.95	85.74	**90.73**
	Macro-F1	43.47	25.35	**70.1**	57.32	69.74
DBLP	Micro-F1	73.28	63.62	63.47	71.34	**74.68**
	Macro-F1	43.24	28.19	20.83	41.62	**49.46**



(a) CiteSeer:Micro-F1 (b) CiteSeer:Macro-F1 (c) DBLP:Micro-F1 (d) DBLP:Macro-F1

Fig. 4. Comparison of BI-FedGNN performance using CiteSeer and DBLP datasets as examples.

4.2. Node classification

A performance evaluation was conducted on the node classification task of BI-FedGNN for federated graph learning on the above five datasets, and results are presented in Table 2. In this study, the Industrial IoT node classification task of federated graph learning was simulated between a central server and four clients. During the experiment, the dataset was randomly assigned to four clients for local model training. Based on the above results, it can be concluded that:

- On the majority of datasets, BI-FedGNN has demonstrated promising experimental results, demonstrating the efficacy of incorporating BI into federated graph neural networks. Using the Bayesian inference approach, clients involved in training can obtain better graph structures, thereby enhancing the efficacy of local GNN model training for clients.
- The BI-FedGNN proposed in this paper has significant performance advantages over the conventional GCN-based FGL. This phenomenon is consistent with predicted outcomes, i.e., if there are noisy edges or lacking strong relationship edges in the graph trained by the model, it will reduce GCN's ability to aggregate accurate information. This indicates that BI-FedGNN can manage the impact of noise and other factors on federated graph learning more effectively.
- In BI-FedGNN, the weights and biases in the client local model training process are randomly selected and optimized as the model iterates. Experimental results show that the randomly selected weights and biases increase the uncertainty of the model and do not have a negative impact on model training. Even in heterogeneous datasets, BI-FedGNN can still achieve good results.

4.3. Performance comparison

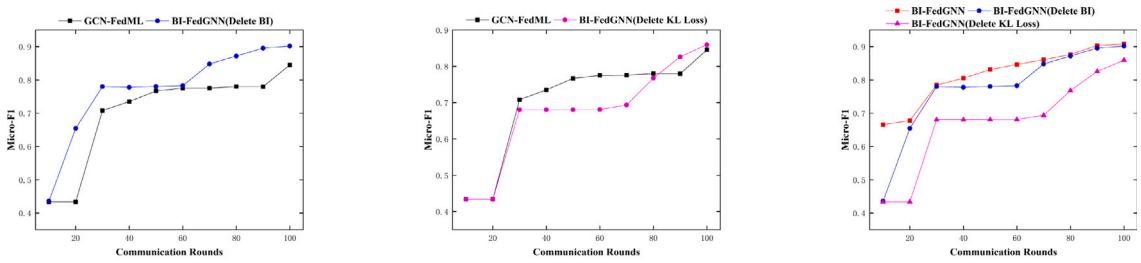
The performance of BI-FedGNN is compared with the rest of the benchmarks using the homogeneous dataset CiteSeer and the heterogeneous dataset DBLP as examples, and the specific results are shown in Fig. 4, which contains the metric curves of Micro-F1 and Macro-F1 for classification. It can be concluded that the performance of BI-FedGNN is always at a higher level in both homogeneous and heterogeneous datasets, which indicates that the method proposed in this paper is well optimized for the classification task of FGL and improves the overall classification accuracy. As shown in the illustration, the accuracy of BI-FedGNN is higher than the rest of the baseline in the early stages of the classification task. This demonstrates that Bayesian inference can well enable the client to obtain better graph structure data for local GNN model training and improve the effectiveness of GNN model training. Meanwhile, the random weights and biases added in the GNN model training process can well improve the uncertainty of the model, making BI-FedGNN can reduce the impact of noise or strong relational edges missing on the overall model training from the beginning of training.

4.4. Ablation analysis

To further validate the optimization brought about by Bayesian inference and KL divergence loss in BI-FedGNN for overall model training, ablation experiments are conducted on BI-FedGNN in this abstract, using the traditional GCN-based federal graph learning as a comparison, with the small dataset CiteSeer serving as the test sample and Micro-F1 and Macro-F1 as the metrics. The specific outcomes are depicted in Figs. 5 and 6. In this study, the BI-FedGNN is divided into three distinct types.

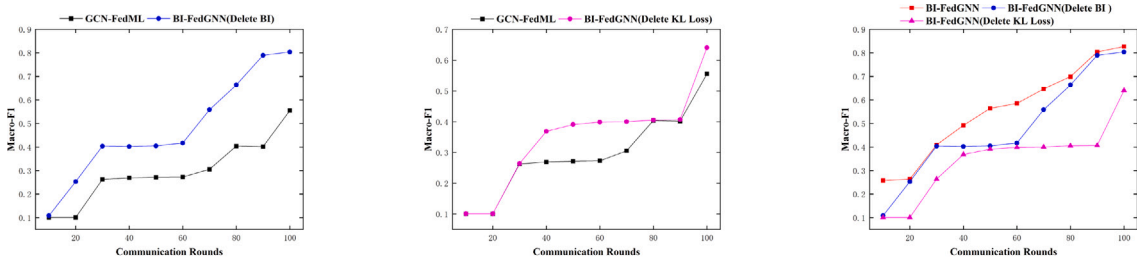
- BI-FedGNN-B: Removes the Bayesian GNN model used in the client's local model training process and instead uses the traditional GCN model to train the client's local data. But keep KL-Loss as part of the overall loss function.
- BI-FedGNN-K: Remove the KL-Loss loss function from the overall loss function, leaving only NLL-Loss. However, the client continues to use the Bayesian GNN model for training during local model training.
- BI-FedGNN: A complete version of the federated graph learning paradigm. The client uses the Bayesian GNN model for training, and also constrains KL-Loss as part of the overall loss function.

Figs. 5 and 6 show the comparison of the three BI-FedGNNs with Micro-F1 and Macro-F1 as metrics with the traditional GCN-based



(a) Delete BI (b) Delete KL Loss (c) BI-FedGNN

Fig. 5. Ablation experiment with Micro-F1 as an indicator.



(a) Delete BI (b) Delete KL Loss (c) BI-FedGNN

Fig. 6. Ablation experiment with Macro-F1 as an indicator.

federal graph learning, respectively. From Figs. 5(a) and 6(a), although BI-FedGNN (Delete BI) is able to achieve good training results, there is a flat or even decreasing trend in the training process. This is due to the fact that although the KL divergence loss function regularizes the training against the overfitting problem, the model lacks the capability of model uncertainty, making it difficult to have effective handling against the missing problem of noisy edges and strongly related edges. Figs. 5(b) and 6(b) show that although BI-FedGNN (Delete KL Loss) has the ability to regularize the model uncertainty, without the KL divergence loss function to regularize the constraint, it will lead to the overfitting problem of the model and even make the training result lower than the unimproved federal graph learning. Both Figs. 5(c) and 6(c) show the full version of BI-FedGNN compared with BI-FedGNN (Delete BI) and BI-FedGNN (Delete KL Loss), which fully demonstrate that Bayesian inference and KL divergence loss function are indispensable for BI-FedGNN. Bayesian inference is used to improve the model uncertainty, while the KL divergence loss function is used as a regularization term to constrain the model and avoid overfitting problems. The two complement each other to improve the classification accuracy of the overall framework.

4.5. Visualization

To verify the effectiveness of the designed framework further, the Coauthor-Physics dataset is used as an example for the low-dimensional vector visualization task, and the BI-FedGNN is contrasted intuitively with the traditional FGL based on GCN, GAT, and SAGE. Specifically, the node embedding vector on the last layer of the hidden layer of the last iteration of the client involved in the training is used to convert the high-dimensional node embedding into a low-dimensional representation using the $t-SNE$ method (Van der Maaten & Hinton, 2008), while the $plt.show()$ method is used to visualize the display. The details are shown in Fig. 7.

In Fig. 7, the results of GCN-FedML, GAT-FedML, and SAGE-FedML are not always satisfactory due to the fact that the subgraphs owned by the clients involved in the training of the industrial Internet may have a gap with the actual graph structure data. This results in the training process receiving noise or missing edges, which in turn causes the degree of differentiation between nodes in different categories to not be obvious. BI-FedGNN performs better in terms of visualization, the classification results are more accurate, and it demonstrates that the nodes of different categories are near one another, with distinct boundaries between them.

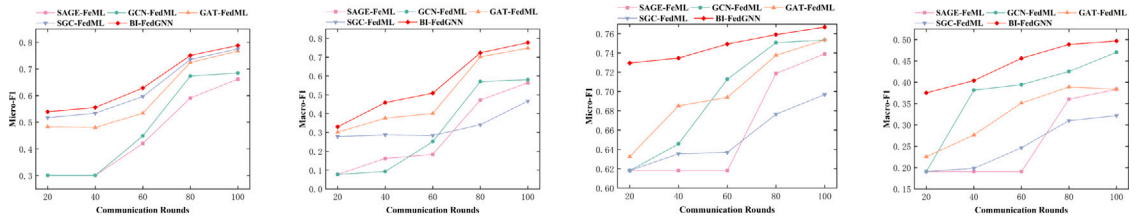
4.6. Parameter analysis

In this paper, the data are processed using the FedML (He et al., 2021) algorithm to self-sample the central nodes, which are being constructed $k-hop$ neighborhoods. Where $k=1, 2, 3$, these datasets are also partitioned and then distributed to the various clients involved in the training. Considering the workload of node sampling, in this article, the maximum sampling hop of the test node is set to 3, because the neighborhood information of 3 hops is enough for the training of the target node. At the same time, it is very time-consuming to sample neighbor information of more than 3 hops for each node in the data set. Again, this subsection takes the homogeneous dataset CiteSeer and the heterogeneous dataset DBLP as examples and focuses on the k-parameters in $k-hop$. This work was tested using the $k=2$ case earlier, so the data comparison graph for the $k=2$ case is not shown in this session; please see Section 4.3 for details.

The comparison of results is shown in Figs. 8 and 9. Please see Table 3 for specific results. Figs. 8 and 9 show the comparison of Micro-F1 and Macro-F1 for the datasets CiteSeer and DBLP for $k=1$ and $k=3$, respectively. With the combination of Fig. 8, Fig. 9 and Table 3, it can be concluded that the classification performance of BI-FedGNNd has been in a relatively advantageous position, and the results show that BI-FedGNN works well with or without the inclusion of higher-order neighbor information, and the performance of BI-FedGNN may become better with the inclusion of more higher-order information. With the increase in higher-order information, BI in BI-FedGNN can better combine additional information node features to process and improve the uncertainty of the model, while KL divergence loss regularizes the loss function to prevent the problem of overfitting. However, the overall running duration will increase as the number of training data points increases.

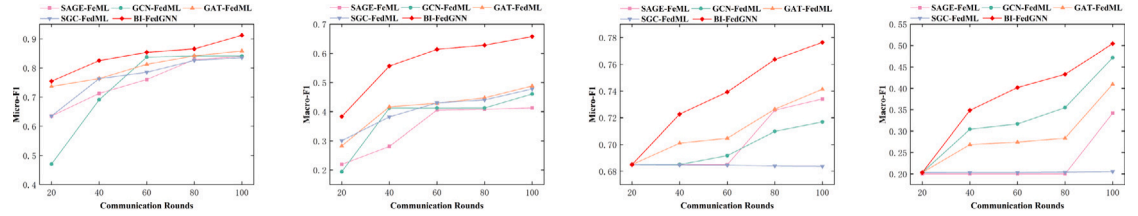
(a) GCN (b) GAT (c) SAGE (d) BI-FedGNN

Fig. 7. Visual representation of the training results of the Coauthor-Physics dataset.



(a) CiteSeer:Micro-F1 (b) CiteSeer:Macro-F1 (c) DBLP:Micro-F1 (d) DBLP:Macro-F1

Fig. 8. Parameter Analysis k=1.



(a) CiteSeer:Micro-F1 (b) CiteSeer:Macro-F1 (c) DBLP:Micro-F1 (d) DBLP:Macro-F1

Fig. 9. Parameter Analysis k=3.

Table 3
Parameter Analysis results (%). (Index: Micro-F1 Macro-F1 Bold: best.)

Datasets	k-hop	Metrics	GCN-FedML	SAGE-FedML	SGC-FedML	GAT-FedML	BI-FedGNN
CiteSeer	1	Micro-F1	68.47	66.19	77.56	76.7	**78.84**
		Macro-F1	58.03	56.4	46.67	74.81	**77.78**
	3	Micro-F1	84.11	84	83.57	85.86	**91.2**
		Macro-F1	46.07	41.29	47.82	48.82	**65.77**
DBLP	1	Micro-F1	75.36	73.91	69.68	75.36	**76.68**
		Macro-F1	47.03	38.36	32.17	38.36	**49.69**
	3	Micro-F1	71.69	73.41	68.37	74.14	**77.64**
		Macro-F1	47.03	38.36	32.17	38.36	**49.69**

4.7. Complexity analysis

In this section, the time complexity of the BI-FedGNN proposed in this article is analyzed. There are n clients participating in the training, the amount of data owned by each client is set to m, and the number of communication rounds during the entire training process is r. In the client-side local model training phase, each data node is subjected to a convolution operation with a time complexity of $O(m \log m)$. Meanwhile, the nodes are sampled with random weights and biases using Bayesian inference before and after the convolution, which in turn has a time complexity of $O(m)$. The time complexity of the two samples is $O(2\ m)$. The time complexity of local training for one client is $O(m(2 + \log m))$, and the complexity of local model training for two clients is $O(n(m(2 + \log m)))$. At the same time, after the client's local model training is completed, the parameters are uploaded to the server for aggregation. A total of r rounds of communication are performed, and the server aggregates local parameters r times in total. Therefore, the overall time complexity of BI-FedGNN is $O(nr^2(m(2 + \log m)))$.

5. Conclusion and future work

This paper investigates and exhibits the effectiveness of adding Bayesian inference to the training of a local client model for federal graph learning. To implement the framework, a Bayesian-based GNN model is designed to adapt to environments where noisy or strongly related edges are missing. Compared with the traditional federal graph learning method, this framework can improve the classification accuracy and robustness of BI-FedGNN by randomly sampling the weights and bias values that obey Gaussian distributions to process the node features of the model, and using KL divergence loss as the regularization term of the overall loss function to prevent the overfitting of the model training. The effectiveness of the BI-FedGNN proposed in this paper is demonstrated by a large number of experiments.

In this paper, the differential privacy (DP) strategy is still used for data transfer from the client to the server. Due to the limited workload, the data protection aspect is not well optimized, and future research will conduct new research on data privacy protection in the process of federal graph learning. During the same time, in the actual federal graph learning process, each client uploads data parameters with different influence and how to realize the server in the parameter aggregation of different clients' parameters according to different weights for different degrees of learning, and whether the trained model of the participating clients can benefit the clients who see the training becomes an urgent problem to be studied.

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.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62272405, School and Locality Integration Development Project of Yantai City(2022), China, the Youth Innovation Science and Technology Support Program of Shandong Provincial, China under Grant 2021KJ080, the Natural Science Foundation of Shandong Province, China under Grant ZR2022MF238, Yantai Science and Technology Innovation Development Plan Project, China under Grant 2022XDRH023.

References

Casiraghi, G., & Nanumyan, V. (2021). Configuration models as an urn problem. *Scientific Reports*, *11*(1), 13416.

Chang, Y., Zhou, W., Cai, H., Fan, W., Hu, L., & Wen, J. (2023). Meta-relation assisted knowledge-aware coupled graph neural network for recommendation. *Information Processing & Management*, *60*(3), Article 103353. http://dx.doi.org/10.1016/j.ipm.2023.103353, URL: https://www.sciencedirect.com/science/article/pii/S0306457323000900.

Chen, F., Long, G., Wu, Z., Zhou, T., & Jiang, J. (2022). Personalized federated learning with a graph. In L. D. Raedt (Ed.), *Proceedings of the thirty-first international joint conference on artificial intelligence* (pp. 2575–2582). International Joint Conferences on Artificial Intelligence Organization, http://dx.doi.org/10.24963/ijcai.2022/357, Main Track.

Chen, C., Zhou, J., Zheng, L., Wu, H., Lyu, L., Wu, J., et al. (2022). Vertically federated graph neural network for privacy-preserving node classification. In L. D. Raedt (Ed.), *Proceedings of the thirty-first international joint conference on artificial intelligence* (pp. 1959–1965). International Joint Conferences on Artificial Intelligence Organization, http://dx.doi.org/10.24963/ijcai.2022/272, Main Track.

Ding, Y., Zhang, Z., Zhao, X., Hong, D., Cai, W., Yang, N., et al. (2023). Multi-scale receptive fields: Graph attention neural network for hyperspectral image classification. *Expert Systems with Applications*, *223*, Article 119858.

Fang, X., & Ye, M. (2022). Robust federated learning with noisy and heterogeneous clients. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10072–10081).

Franceschi, L., Niepert, M., Pontil, M., & He, X. (2019). Learning discrete structures for graph neural networks. In *International conference on machine learning* (pp. 1972–1982). PMLR.

Fu, X., Zhang, B., Dong, Y., Chen, C., & Li, J. (2022). Federated graph machine learning: A survey of concepts, techniques, and applications. *ACM SIGKDD Explorations Newsletter*, *24*(2), 32–47.

Gaudelet, T., Day, B., Jamasb, A. R., Soman, J., Regep, C., Liu, G., et al. (2021). Utilizing graph machine learning within drug discovery and development. *Briefings in Bioinformatics*, *22*(6), bbab159.

Giles, C. L., Bollacker, K. D., & Lawrence, S. (1998). CiteSeer: An automatic citation indexing system. In *Proceedings of the third ACM conference on digital libraries* (pp. 89–98).

He, C., Balasubramanian, K., Ceyani, E., Yang, C., Xie, H., Sun, L., et al. (2021). Fedgraphnn: A federated learning benchmark system for graph neural networks. In *ICLR 2021 workshop on distributed and private machine learning*.

He, C., Ceyani, E., Balasubramanian, K., Annavaram, M., & Avestimehr, S. (2022). Spreadgnn: Decentralized multi-task federated learning for graph neural networks on molecular data. In *Proceedings of the AAAI conference on artificial intelligence, vol. 36, no. 36* (pp. 6865–6873).

Jiang, B., Zhang, Z., Lin, D., Tang, J., & Luo, B. (2019). Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11313–11320).

Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., & Tang, J. (2020). Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 66–74).

Jing, C., Huang, Y., Zhuang, Y., Sun, L., Xiao, Z., Huang, Y., et al. (2023). Exploring personalization via federated representation learning on non-IID data. *Neural Networks*, *163*, 354–366. http://dx.doi.org/10.1016/j.neunet.2023.04.007, URL: https://www.sciencedirect.com/science/article/pii/S0893608023001843.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., et al. (2021). Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, *14*(1–2), 1–210.

Li, X., JIANG, M., Zhang, X., Kamp, M., & Dou, Q. (2021). FedBN: Federated learning on non-IID features via local batch normalization. In *International conference on learning representations*. URL: https://openreview.net/forum?id=6YEQUn0QICG.

Li, R., Liu, Z., Ma, Y., Yang, D., & Sun, S. (2022). Internet financial fraud detection based on graph learning. *Ieee Transactions on Computational Social Systems*.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated optimization in heterogeneous networks. In *Proceedings of machine learning and systems, vol. 2* (pp. 429–450).

Liang, W., Chen, X., Huang, S., Xiong, G., Yan, K., & Zhou, X. (2023). Federal learning edge network based sentiment analysis combating global COVID-19. *Computer Communications*, *204*, 33–42.

Liu, Y., Ding, K., Liu, H., & Pan, S. (2023). GOOD-D: On unsupervised graph out-of-distribution detection. In *Proceedings of the sixteenth ACM international conference on web search and data mining* (pp. 339–347).

Liu, Z., Yang, D., Wang, Y., Lu, M., & Li, R. (2023). EGNN: Graph structure learning based on evolutionary computation helps more in graph neural networks. *Applied Soft Computing*, Article 110040.

Liu, Z., Yang, D., Wang, S., & Su, H. (2022). Adaptive multi-channel Bayesian graph attention network for IoT transaction security. *Digital Communications and Networks*.

McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics* (pp. 1273–1282). PMLR.

Meng, C., Rambhatla, S., & Liu, Y. (2021). Cross-node federated graph neural network for spatio-temporal data modeling. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (pp. 1202–1211).

Pei, Y., Mao, R., Liu, Y., Chen, C., Xu, S., Qiang, F., et al. (2021). Decentralized federated graph neural networks. In *International workshop on federated and transfer learning for data sparsity and confidentiality in conjunction with IJCAI*.

Peng, L., Wang, N., Dvornek, N., Zhu, X., & Li, X. (2022). Fedni: Federated graph learning with network inpainting for population-based disease prediction. *IEEE Transactions on Medical Imaging*.

Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., et al. (2020). Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1150–1160).

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29*(3), 93.

Shchur, O., Mumme, M., Bojchevski, A., & Günnemann, S. (2018). Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868.

Shen, X., Pan, S., Choi, K.-S., & Zhou, X. (2023). Domain-adaptive message passing graph neural network. *Neural Networks*, *164*, 439–454. http://dx.doi.org/10.1016/j.neunet.2023.04.038, URL: https://www.sciencedirect.com/science/article/pii/S0893608023002253.

Shu, J., Yang, T., Liao, X., Chen, F., Xiao, Y., Yang, K., et al. (2023). Clustered federated multitask learning on non-IID data with enhanced privacy. *IEEE Internet of Things Journal*, *10*(4), 3453–3467. http://dx.doi.org/10.1109/JIOT.2022.3228893.

Sun, J., Gao, L., Shen, X., Liu, S., Liang, R., Du, S., et al. (2023). Separated graph neural networks for recommendation systems. *IEEE Transactions on Industrial Informatics*, *19*(1), 382–393. http://dx.doi.org/10.1109/TII.2022.3194659.

Tan, Y., Liu, Y., Long, G., Jiang, J., Lu, Q., & Zhang, C. (2022). Federated learning on non-IID graphs via structural knowledge sharing. arXiv preprint arXiv:2211.13009.

Tang, C., Li, Z., Wang, J., Liu, X., Zhang, W., & Zhu, E. (2022). Unified one-step multi-view spectral clustering. *IEEE Transactions on Knowledge and Data Engineering*, *35*(6), 6449–6460.

Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., & Su, Z. (2008). Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 990–998).

Tang, C., Zheng, X., Zhang, W., Liu, X., Zhu, X., & Zhu, E. (2023). Unsupervised feature selection via multiple graph fusion and feature weight learning. *Science China. Information Sciences*, *66*(5), 1–17.

Tun, Y. L., Nguyen, M. N., Thwal, C. M., Choi, J., & Hong, C. S. (2023). Contrastive encoder pre-training–based clustered federated learning for heterogeneous data. *Neural Networks*, http://dx.doi.org/10.1016/j.neunet.2023.06.010, URL: https://www.sciencedirect.com/science/article/pii/S0893608023003192.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(11).

Wan, Y., Yuan, C., Zhan, M., & Chen, L. (2022). Robust graph learning with graph convolutional network. *Information Processing & Management*, *59*(3), Article 102916.

Wang, K., An, J., Zhou, M., Shi, Z., Shi, X., & Kang, Q. (2023). Minority-weighted graph neural network for imbalanced node classification in social networks of internet of people. *IEEE Internet of Things Journal*, *10*(1), 330–340. http://dx.doi.org/10.1109/JIOT.2022.3200964.

Wang, Y., Liu, Z., Xu, J., & Yan, W. (2022). Heterogeneous network representation learning approach for ethereum identity identification. *IEEE Transactions on Computational Social Systems*.

Wang, R., Mou, S., Wang, X., Xiao, W., Ju, Q., Shi, C., et al. (2021). Graph structure estimation neural networks. In *Proceedings of the web conference 2021* (pp. 342–353).

Wang, W., Wang, Y., Duan, P., Liu, T., Tong, X., & Cai, Z. (2022). A triple real-time trajectory privacy protection mechanism based on edge computing and blockchain in mobile crowdsourcing. *IEEE Transactions on Mobile Computing*.

Wu, S., Sun, F., Zhang, W., Xie, X., & Cui, B. (2022). Graph neural networks in recommender systems: A survey. *ACM Computing Surveys*, *55*(5), 1–37.

Wu, C., Wu, F., Lyu, L., Qi, T., Huang, Y., & Xie, X. (2022). A federated graph neural network framework for privacy-preserving personalization. *Nature Communications*, *13*(1), 3091.

Xie, H., Ma, J., Xiong, L., & Yang, C. (2021). Federated graph classification over non-iid graphs. In *Advances in neural information processing systems, vol. 34* (pp. 18839–18852).

Xie, Z., & Tu, Y. (2022). A graph convolutional network with adaptive graph generation and channel selection for event detection. In *Proceedings of the AAAI conference on artificial intelligence, vol. 36, no. 10* (pp. 11522–11529).

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? In *International conference on learning representations*. URL: https://openreview.net/forum?id=ryGs6iA5Km.

Yan, W., Gu, M., Ren, J., Yue, G., Liu, Z., Xu, J., et al. (2023). Collaborative structure and feature learning for multi-view clustering. *Information Fusion*, *98*, Article 101832.

Zhang, H., Shen, T., Wu, F., Yin, M., Yang, H., & Wu, C. (2021). Federated graph learning–A position paper. arXiv preprint arXiv:2105.11099.

Zhou, S., Song, P., Yu, Y., & Zheng, W. (2023). Structural regularization based discriminative multi-view unsupervised feature selection. *Knowledge-Based Systems*, *272*, Article 110601.