Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.060331

ARTICLE

Decentralized Federated Graph Learning via Surrogate Model

Bolin Zhang, Ruichun Gu* and Haiying Liu

School of Digital and Intelligence Industry (School of Cyber Science and Technology), Inner Mongolia University of Science and Technology, Baotou, 014010, China

*Corresponding Author: Ruichun Gu. Email: reachcool@imust.edu.cn

Received: 29 October 2024 Accepted: 22 November 2024 Published: 17 February 2025

ABSTRACT

Federated Graph Learning (FGL) enables model training without requiring each client to share local graph data, effectively breaking data silos by aggregating the training parameters from each terminal while safeguarding data privacy. Traditional FGL relies on a centralized server for model aggregation; however, this central server presents challenges such as a single point of failure and high communication overhead. Additionally, efficiently training a robust personalized local model for each client remains a significant objective in federated graph learning. To address these issues, we propose a decentralized Federated Graph Learning framework with efficient communication, termed Decentralized Federated Graph Learning via Surrogate Model (SD_FGL). In SD_FGL, each client is required to maintain two models: a private model and a surrogate model. The surrogate model is publicly shared and can exchange and update information directly with any client, eliminating the need for a central server and reducing communication overhead. The private model is independently trained by each client, allowing it to calculate similarity with other clients based on local data as well as information shared through the surrogate model. This enables the private model to better adjust its training strategy and selectively update its parameters. Additionally, local differential privacy is incorporated into the surrogate model training process to enhance privacy protection. Testing on three real-world graph datasets demonstrates that the proposed framework improves accuracy while achieving decentralized Federated Graph Learning with lower communication overhead and stronger privacy safeguards.

KEYWORDS

Federated learning; federated graph learning; decentralized; graph neural network; privacy preservation

1 Introduction

With the rapid advancement of artificial intelligence and big data technologies, a substantial amount of data has been accumulated. Since graph models can effectively represent social relationships, transportation networks, knowledge databases, and more, the volume of graph data is also growing exponentially. This graph data conceals a wealth of potential information, which holds significant research value. In the analysis and processing of graph data, the ideal scenario is for a party to have complete ownership of the entire graph dataset, enabling direct execution of graph task learning and associative reasoning. However, in practical application environments, a complete graph dataset is typically composed of multiple subgraphs, which are often stored in the local systems of

various terminals [1]. Due to concerns regarding data security and privacy, the graph data held by each party is protected and not directly shared, preventing any terminal from accessing certain graph data owned by other parties. Consequently, Graph Neural Networks (GNNs) [2] can only train on the local subgraph data of each party, which significantly hampers the model's generalization ability.

As a revolutionary paradigm, Federated Learning (FL) [3] can collaboratively train a global model in a distributed manner without sharing clients' local data, thereby ensuring data privacy and security. This approach effectively mitigates the risk of privacy leakage associated with the direct exchange of data between users. For local clients with limited training data and insufficient data diversity, FL allows for data support from multiple clients, significantly enhancing the performance of local task training and improving the model's generalization ability. Due to its advantages, FL can be effectively applied to graph data, addressing the issue of graph islands while safeguarding the security of graph data owned by end users. Consequently, FL demonstrates excellent performance when utilized in graph learning tasks. However, in traditional FL, all parameters are aggregated and updated on a central server, which then distributes the updated parameters to each client. This process incurs significant communication overhead. Additionally, the reliance on a central server introduces security vulnerabilities. On one hand, a single central server is inherently fragile and susceptible to attacks, creating a potential single point of failure. On the other hand, the possibility of malicious behavior from the central server itself cannot be dismissed. During model training, a compromised central server could exploit user information, leading to privacy breaches. Currently, high communication overhead and central server security represent two critical challenges in centralized federated learning. For instance, it is unreliable for multiple hospitals to collaborate in establishing a central server, as each hospital possesses its own local, non-independent and identically distributed (non-IID) dataset. Each institution aims to train a model that is more attuned to its specific local data while selectively adjusting the training strategy based on its unique expertise. This situation may result in diminished training performance.

To address these challenges, researchers have approached the problem from various perspectives, including the use of blockchain technology to achieve decentralization [4]. However, existing decentralized FGL methods struggle to ensure high accuracy for each client model while accommodating the personalized requirements of individual clients and maintaining low communication overhead. In response to this issue, this paper proposes a Decentralized Federated Graph Learning via Surrogate Model, building on previous research. The proposed method, referred to as SD_FGL, trains two models locally: a surrogate model that shares parameters and gradient information with other clients on a one-to-one basis, and a private model that adjusts its training strategy based on insights gained from the surrogate model. This framework enables decentralized FGL, allowing each client to train a personalized and efficient model without compromising communication efficiency or privacy. The contributions of this paper are as follows:

1. To address the challenges associated with a central server, we propose a decentralized federated graph learning framework, SD_FGL, which offers high communication efficiency. Each client maintains two models: a private model for local use and a surrogate model for information sharing among clients.

2. The surrogate model incorporates local differential privacy into data transmission to safeguard the privacy of client data. The private model computes similarity based on the information from the surrogate model shared by other clients and updates it according to the similarity weights. This dual model effectively preserves privacy while ensuring accuracy.

3. Extensive experiments conducted on real datasets demonstrate that our framework has lower communication overhead while maintaining accuracy compared to existing baselines.

2 Related Work

2.1 Federated Learning

Federated learning (FL) has garnered increasing attention as an emerging distributed machine learning paradigm that facilitates collaborative training without the need to share raw data. The primary objective of FL is to safeguard data privacy and security while enhancing the performance and generalization capabilities of the model by leveraging distributed data resources. FedAvg is the original standard FL framework, which first trains a model independently at each client, subsequently performs global parameter averaging at a central server, and then transmits the aggregated model back to the clients. Since then, several centralized FL frameworks have been developed, including FATE [5], TensorFlow Federated (TFF) [6], and PySyft [7]. However, these centralized FL frameworks continue to face challenges related to communication and the security of the central server.

In order to address the security challenges associated with centralized FL frameworks, decentralized FL frameworks have garnered significant attention. Roy et al. [8] proposed a peer-to-peer decentralized FL algorithm, while Lalitha et al. [9] explored a fully decentralized FL algorithm, and group learning has utilized blockchain technology to enhance decentralized and secure collaborative training networks. In each round of voting, one client is elected as the central authority. Swarm learning does not alter the core learning algorithm of FL; therefore, it inherits relatively poor model performance when stringent privacy measures are applied and necessitates homogeneous model architectures. Although these implementations are decentralized, the performance and communication efficiency of the models trained within the framework require improvement. When confronted with heterogeneous or non-IID data, the accuracy of the models is significantly compromised.

In order to address the communication efficiency challenges associated with the centralized FL framework, researchers have primarily concentrated their efforts on model compression techniques to alleviate communication burdens. Hu et al. [10] employed the Gossip algorithm to enhance bandwidth utilization and mitigate communication pressure through model segmentation. Amiri et al. [11] proposed model quantization methods to further reduce communication demands.

To enhance the performance of FL on non-IID datasets, current research primarily emphasizes centralized FL approaches. FedProx [12] introduces a regularization term designed to minimize the weight disparity between the local model and the global model, thereby preventing the local model from becoming overly tailored to the local training data. FedPer [13] presents a method for sharing the base layer when the non-IID phenomenon is particularly pronounced, while also providing a personalized layer for each client to preserve local knowledge. Compared to the global model, the personalized model is more suitable for individual clients. In the context of decentralized FL, Li et al. [14] proposed a targeted decentralized FL framework to address non-IID data; however, its performance is inferior to that of FedAvg in scenarios without data poisoning.

Although numerous frameworks have been proposed to address various challenges in FL, they often fail to simultaneously consider issues such as high communication overhead, central server security, and privacy protection in non-IID scenarios.

2.2 Federated Graph Learning

Federated Graph Learning (FGL) facilitates distributed training of GNNs by employing a FL framework that enables co-training of GNNs without the need to share client graph data, thereby broadening the original application scenarios of FL. Current research in FGL can be categorized into three primary areas [15]: inter-graph federated graph learning, intra-graph federated graph learning,

and graph-structured federated graph learning. In inter-graph federated learning, each client possesses a set of completely disjoint graphs and participates in FL training while addressing the non-IID problem to develop a more effective local GNN model. In intra-graph federated learning, each client owns only a subgraph of a complete global graph. The primary challenge in this scenario is the potential for missing links between subgraphs. Existing methodologies [1] enhance nodes by soliciting information from other subgraphs to generate these missing links; however, this process may compromise data privacy. In graph structure federated learning, graph data is organized through the ego network, which represents the internal relationships between clients and is applied to various datasets, such as Fedsage+ [1], FedGCN [16], GCFL [17]. Within the realm of GNN applications, the necessity for processing distributed graph data is becoming increasingly pronounced. In response to this need, Pei et al. [18] introduced Decentralized Federated Graph Neural Networks (DFGNN), which amalgamates the advantages of decentralized federated learning with graph neural networks. This approach employs a two-stage strategy: initially, graph neural network training is conducted locally on the data of each participant to generate model updates. Subsequently, the exchange and aggregation of updated parameters among neighboring participants occurs in a decentralized manner, thereby eliminating dependence on a central server. Currently, the majority of existing studies concentrate on model accuracy with non-IID graph data within a centralized federated graph learning (FGL) framework. However, they frequently neglect the substantial communication overhead and privacy protection challenges associated with a centralized server.

2.3 Differential Privacy in FL

Although the original data never leaves the local client, privacy violations can still occur in FL. Differential privacy (DP) has been integrated with FL, and some literature employs DP to safeguard data privacy. Reference [19] presents a GNN-based privacy-preserving learning framework that offers formal privacy guarantees based on edge local differential privacy. The key innovation lies in a new set of sophisticated mechanisms designed to calibrate the noise introduced by the scatter maps collected from users. FedGNN operates within a two-server model. In Reference [20], the authors propose a vertically federated GNN learning model (VFGNN) for privacy-preserving node classification tasks. This model assigns the computation related to private data to the data holder while delegating the remaining computations to a semi-honest server. However, the framework proposed in this study assumes that each client possesses the same nodes, which is not reflective of reality. For instance, the data held by major hospitals are non-IID, significantly diminishing the efficiency of the proposed framework in practical applications.

In this paper, we propose a federated graph learning framework designed to enhance the security of the central server while addressing the issue of high communication overhead. This framework achieves decentralization by allowing local clients to maintain both a private model and a surrogate model, with the surrogate model facilitating communication between clients. Additionally, we incorporate local differential privacy during the training of the surrogate model. The local private model can compute similarity based on the received surrogate model, enabling the training of a more robust private model. Since the text frame employs one-to-one message passing and the parameters of the surrogate model are minimal, the communication costs are significantly reduced.

3 Definitions and FaS-Fed Design Implementation

3.1 Definition of Graph Neural Networks

Generally speaking, a graph consists of a set of n nodes and a set of m edges, with each node possessing a feature vector. The goal of GNNs is to learn the representation of the graph based on its structure and the features of its nodes. For instance, GNNs can learn a node-level representation vector for individual nodes or a graph-level representation vector for the entire graph. Existing GNNs typically employ a message-passing approach, where each node iteratively collects information propagated from its neighbors. Subsequently, it updates its representation by aggregating its own information with the collected data, effectively refining its representation by incorporating the representations of neighboring nodes. For an L-layer GNN, the representation of the L-th layer is defined as follows:

$$a_{\mathcal{V}}^{(l)} = AGG^{(l)}\left(\left\{h_u^{(l-1)} : \forall u \in \mathcal{N}(v)\right\}\right) \tag{1}$$

$$h_v^{(l)} = \mathrm{UPD}^{(l)}\left(h_v^{(l-1)}, a_{\mathcal{V}}^{(l)}\right) \tag{2}$$

where $h_v^{(l)}$ is the feature representation vector of node v's output at layer L, where $h_v^{(0)} = x_v$ is initialized as the node feature, and $\mathcal{N}(v)$ is the set of adjacent nodes of node v: $\mathcal{N}(v) = \{u \in V | (u, v) \in E\}$. The AGG aggregates the neighbor features of node v, and the UPD updates the neighbor features of node v based on the previous representation of node v and the aggregated neighbor representation of node v. Different GNNS can adopt different AGG and UPD policies.

Finally, summing, pooling and mean pooling can be performed by the readout function to aggregate the representation of all nodes into a single embedded vector, thus realizing the graph classification task.

3.2 Definition of Federated Learning

The goal of FL is to train a global model using local proprietary data. The foundational FL algorithm is the Stochastic Gradient Descent (SGD)-based aggregation algorithm known as FedAvg. Due to the powerful optimization capabilities of SGD, FedAvg is frequently employed as the basis for designing more efficient FL frameworks. The core concept of FedAvg is to aggregate the updated model parameters from local clients and subsequently distribute these aggregated parameters back to each client. The process is outlined as follows: Suppose there are multiple clients, each possessing private data that is inaccessible to other clients. The working principle is as follows: (1) Initialized on round 0 communication, the local model parameters of the K clients are initialized to the global parameter $\overline{\theta}$. (2) Each client utilizes the parameters from the previous round to train on its local data D_s, aiming to minimize task loss, and subsequently updates its local parameters. (3) Once local training is complete, the server collects the updated local parameters from each client, aggregates these parameters by averaging, and then distributes the updated global parameters to the clients that will continue training in the next round. This allows the participating clients to update their local parameters. After the first round is completed, the cycle continues between steps 2 and 3 until the final round of training iterations is reached. At the end of the cycle, each client receives the final model parameters. Throughout this process, the server only updates the parameters in aggregate, without involving the sharing of private data.

3.3 Local Client GNN Network

Nowadays, more federated learning directly shares the feature-based GNN encoder globally. Due to the different domain of each client, the model performance of the client may be reduced when

performing federated learning. Therefore, the federated learning of features in similar domains can be performed while utilizing structural embeddings to capture the common structural information between graphs in different domains, so that each client learns a more robust local model.

Inspired by [21], this paper uses a structure-feature dual-channel GNN, which learns structure knowledge and feature knowledge through two parallel channels. In the structure-based channel, the structure encoder learns on the basis of the structure embedding and propagates the structure information on the global graph. At the same time, in the feature-based channel, the feature encoder learns on the basis of feature embedding by using random graphs as input to propagate the information of features on similar domain graphs. This GNN model consists of three parts, (i) h_f: a linear layer with learnable parameters ω_s; (ii) h_f and h_s: a GNN layer with structure-feature dual channels stacked by a three-layer graph convolutional network, this layer is parameterized as ω_s and ω_f; (iii) a linear classifier layer for node classification.

Given an input graph $G = (V; E)$, assume that a node $v \in V$ has a feature vector x_v and a structure vector s_v. In the first step, through a linear layer, s_v is transformed into the desired input embedding, and x_v is not manipulated. Through this linear layer, the input of the structural channel is transformed into a vector of the specified dimension.

Then, three layers of graph convolutional layers are entered, one layer is used for structure knowledge learning, and the other two layers are used for feature knowledge learning. In the structure encoder, the structure vector of the specified dimension is output by performing the aggregate update. At this time, the structure encoder is only used to learn the general structure knowledge, and will not obtain any signal as input in the feature knowledge. Meanwhile, the feature encoder part, based on the random graph as input, first passes through the first layer of graph convolutional network to calculate the feature embedding of its local GNN model. These feature embeddings are sent to the specified client and used to calculate the similarity between the clients receiving the information, and a weighted average is performed to aggregate this client surrogate model. In the second layer of the graph convolutional network, the sparse mask of its local GNN model is computed, which is used to selectively update the parameters of similar features with the local domain. Through dual-channel GNN, the structure information and feature information can be learned independently, and the feature learning will not be affected by the feature knowledge of different fields.

After three layers of GNN, we enter the linear classifier layer, which concatenates the output node-level feature embeddings as well as the structure embeddings together. Then, we follow the steps of the node classification task, where the feature parameters are also selectively updated using masks. The first m said a client can learn parameter set for the $\omega_m = \{\omega_{f,m}, \omega_{s,m}\}$ the $\omega_{f,m}$ for the characteristics of the encoder and the parameters in the classifier, $\omega_{s,m}$ for the parameters in the structure of the encoder. Compared with the traditional single-channel GNN, the two-channel GNN used in the proposed framework has more obvious advantages in the non-IID FGL scenario. On the one hand, the structural embedding learned by the structural encoder can project the common structural information of data in different domains into the same representation space, so that the structural knowledge can be shared between different domains without being affected by the features between different domains. On the other hand, the structural encoder can learn the feature knowledge in similar domains. Finally, the output results of the two channels were fused to obtain a more accurate model.

3.4 Local Client Dual Model

In FGL, since the data held by each client is in different professional fields, the models trained between each client are basically heterogeneous, and the central server is basically trained as a global

universal global model, and the central server has a single point of failure problem. Therefore, considering decentralized FGL, in considering the proposed framework, suppose there are M clients, each with its local data D_m, $\forall m \in M$, and each client maintains a private model ω_m. We need to train an efficient and robust model for each client. After the private model is trained, if the private model information is directly sent to other clients, it may expose the private information of the local client, resulting in increased risk of adversarial attacks and increased risk of privacy exposure. In order to solve this problem, we introduce a common surrogate model for each client, whose parameter is φ_m, and each surrogate model is used as an interface with every other client to realize the transmission of parameters. Notably, each local surrogate model shares the same structure as the private model.

In SD_FGL, each client first trains its local private model and its surrogate model, and they can benefit from each other during training. Through differential privacy training, the surrogate model can extract useful information from the private data, which can be shared with other clients without revealing privacy. Each client then sends its surrogate model parameter information to the specified client and receives surrogate model parameters from the specified client. Finally, each client summarized the received surrogate model, calculated the similarity between the received surrogate model and the local surrogate model, and weighted average the parameters of the local surrogate model according to the model similarity, so that each client tended to perform the weighted average of parameters in the relevant domain. This approach allows each client to update parameters that benefit it. The overall process is shown in Fig. 1 and Algorithm 1.

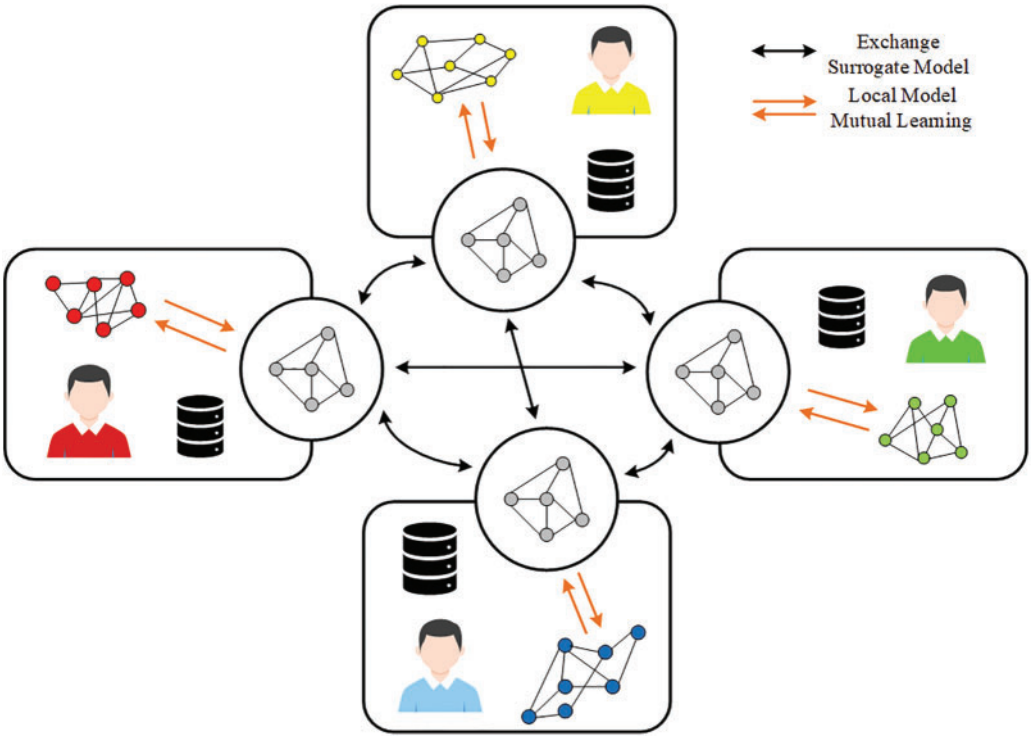

Figure 1: Local client dual model and interaction with other clients

Our proposed framework, applied to classification tasks, trains a private model for each client at the beginning of each round, by adding cross-entropy loss (CE) as well as KL divergence loss (KL) when training the model, so that the private models can learn from each other in the surrogate model.

The two loss functions are as follows:

$$\mathcal{L}_{\mathrm{CE}}\left(f_{\omega_m}\right): = \mathbb{E}_{(x,y)\sim D_m}\mathrm{CE}\left[f_{\omega_m}(x) \parallel y\right] \tag{3}$$

$$\mathcal{L}_{\mathrm{KL}}\left(f_{\omega_m}; h_{\varphi_m}\right): = \mathbb{E}_{(x,y)\sim D_m}\mathrm{KL}\left[f_{\omega_m}(x) \parallel h_{\varphi_m}(x)\right] \tag{4}$$

where f_{ω_m} and h_{φ_m} are the local private model and the surrogate model of client m respectively, and the goal of learning the private model is:

$$\mathcal{L}_{\omega_m}: = (1-\alpha)\cdot\mathcal{L}_{\mathrm{CE}}\left(f_{\omega_m}\right) + \alpha\cdot\mathcal{L}_{\mathrm{KL}}\left(f_{\omega_m}; h_{\varphi_m}\right) \tag{5}$$

The goals of the surrogate model are similarly defined as:

$$\mathcal{L}_{\varphi_m}: = (1-\beta)\cdot\mathcal{L}_{\mathrm{CE}}\left(h_{\varphi_m}\right) + \beta\cdot\mathcal{L}_{\mathrm{KL}}\left(h_{\varphi_m}; f_{\omega_m}\right) \tag{6}$$

where $\alpha, \beta \in (0,1)$ is used to balance between the two losses, and the stochastic gradient step is alternated between the private model and the surrogate model.

Set $\Phi^{(r)} \in \mathbb{R}^{(|M|\times d_\varphi)}$ said the first r round stack surrogate, the line is the surrogate parameter $\varphi_m^{(r)}$, for all $m \in M$. We denote by $P^{(r)} \in \mathbb{R}^{|M|\times|M|}$ the weighted adjacency matrix representing the graph topology at round t, where $P_{m,m'}^{(r)} \neq 0$ means that client m receives a surrogate from client m′. At this time, $P^{(r)}$ only needs random columns without symmetry, and only needs one-way communication, only needs to accept the information of the designated client surrogate model and send the local client surrogate model to other designated clients, which ensures the efficiency of communication. This method can adapt to the client joining or leaving, and can automatically select the client number to send and receive the surrogate model according to the number of clients in each round. During the transmission of the surrogate model, when each client receives the surrogate model parameters from other clients, the parameters are averaged in accordance with the similarity between the information provided by the local client and that of the client transmitting the surrogate model:

$$\overline{\varphi}_m \leftarrow \sum_{j=1}^{K}\alpha_{mj}\cdot\varphi_j, \alpha_{mj} = \frac{\exp\left(\mu\cdot F(m,j)\right)}{\sum_k \exp\left(\mu\cdot F(m,k)\right)} K = \{m, m'\} \tag{7}$$

where α_{mj} is the normalized similarity between the local client m and the received client j, μ is the hyperparameter used to scale the unnormalized similarity measure. As the value of μ increases, each client tends to perform the parameter weighted average in the related domain, that is, similar clients will have more weight. This approach allows each client to update parameters that benefit it.

Algorithm 1: Decentralized federated graph learning model.

Require: the number of rounds R, the number of epochs E, the number of clients M, private parameters ω_m, surrogate parameters φ_m, loss function weights $\alpha, \beta \in (0,1)$, adjacency matrix $P^{(r)}$, learning rate η, scaling factor for similarity matching μ, Feature embedding for calculating similarity F_m.

1 **for** each round $r = 1, 2, \ldots, R$ **do**
2 **for** each local optimization step **do**
3 Sample batch $B_m = \{(x_i, y_i)\}_{i=1}^{N}$ from D_m;
4 Update local private and surrogate models:

$$\omega_m^{(r)} \leftarrow \omega_m^{(r)} - \eta\nabla\mathcal{L}_{\omega_m}(B_m) \quad \text{\# non-LDP update}$$
$$\varphi_m^{(r)} \leftarrow \varphi_m^{(r)} - \eta\nabla\mathcal{L}_{\varphi_m}(B_m)' \quad \text{\#LDP update}$$

5 **end for**
6 $\omega_m^{(r+1)} \leftarrow \omega_m^{(r)}$;

(Continued)

Algorithm 1 (continued)

Send $(P^{(r)}_{m',m}\varphi^{(r)}_{m}, F_m)$ to out-neighbors;
Receive $\leftarrow (P^{(r)}_{m,m'}\varphi^{(r)}_{m'}, F_{m'})$ from out-neighbors;
if $r = 1$ **then**
    Update local surrogate: $\varphi^{(r+1)}_{m} \leftarrow \varphi^{(0)}_{m}$
else
    Update local surrogate:

$$\varphi^{(r+1)}_{m} \leftarrow \sum_{j=1}^{K} \frac{\exp\left(\mu \cdot F\left(m,j\right)\right)}{\sum_{k} \exp\left(\mu \cdot F\left(m,k\right)\right)} \cdot \varphi^{(r)}_{j} \qquad K = \{m, m'\}$$

end if
Update local private: $\omega^{(r+1)}_{m} \leftarrow \omega^{(r)}_{m}$
end for
return $\varphi^{T}_{m}, \omega^{T}_{m}$

3.5 Privacy Protection During Communication

In federated graph learning, each client performs message passing and model parameter sharing with other clients through the surrogate model. To ensure that this sharing does not leak the data privacy of each client, we introduce a differential privacy (DP) mechanism. Differential privacy protects user privacy by adding noise to the data and gradients, obscuring the existence of individual data points. Specifically, the core formula of differential privacy is:

$$Pr[M(D) \in S] \leq exp(\epsilon)Pr[M(D') \in S] + \delta. \tag{8}$$

This formula says that for all possible subsets of outputs (S), the ratio of the probability that mechanism (M) produces some output falling in (S) on databases (D) and (D') differs by at most $exp(\epsilon)$ times, plus a small δ. If the mechanism satisfies the formula conditions, then the mechanism is said to be (ϵ, δ) -differentially private. Even if databases (D) and (D') differ by only one data point, the output distributions of mechanism (M) will not differ significantly. Thus, differential privacy ensures that even if an attacker knows all but one data point in the database, they cannot determine whether this particular data point is present in the database by observing the output of the mechanism (M). In our proposed framework, we achieve local differential privacy (LDP) by the following steps:

Data perturbation: Before each client uploading graph data, node features and edge weights are perturbed. Assuming that the original data is (x), the perturbed data (x') can be expressed as: $x' = x+n$ where the noise n follows a normal distribution $\mathcal{N}\left(0, \sigma^2\right)$, where σ is the standard deviation of the noise and is related to the privacy budget ϵ.

Local training process: We sample a mini-batch $B_m \in D_m$ from the client data, and at each model training update, noise is added to the gradient to protect the privacy of the model parameters. Where each client's local private model gradient is:

$$\nabla \mathcal{L}_{\omega_m}\left(B_m\right) = \frac{1}{N} \sum\nolimits_{i=1}^{N} g^{(i)}_{\omega_m} \tag{9}$$

where:

$$g^{(i)}_{\omega_m} : = (1-\alpha)\nabla_{\omega_m}\mathrm{CE}[f_{\omega_m}(x_i)|\,|y_i] + \alpha\nabla_{\omega_m}\mathrm{KL}[f_{\omega_m}(x_i)|\,|h_{\varphi_m}(x_i)] \tag{10}$$

The same per-client surrogate model with parameters φ_m and gradient $\nabla\mathcal{L}_{\varphi m}(B_m)$ Similar to the private model gradient, the gradient $\nabla\mathcal{L}_{\varphi m}(B_m)'$ after perturbing the surrogate model can be expressed as follows. $\nabla\mathcal{L}_{\varphi m}(B_m)' = \nabla\mathcal{L}_{\varphi m}(B_m) + \mathcal{N}\left(0, \sigma^2\right)$, $\mathcal{N}(0, \sigma^2)$ is to obey the normal distribution of noise, sigma budget ϵ related with privacy.

In turn, the local surrogate model parameters φ_m and the private model parameters ω_m are updated. The specific update process is as follows:

$$\omega_m^{(r)} \leftarrow \omega_m^{(r)} - \eta\nabla\mathcal{L}_{\omega m}(B_m) \tag{11}$$

$$\varphi_m^{(r)} \leftarrow \varphi_m^{(r)} - \eta\nabla\mathcal{L}_{\varphi m}(B_m)' \tag{12}$$

Ultimately, privacy guarantees are tracked on a per-client basis. In multi-agency cooperation, each client is obliged to protect the privacy of its collected data. Thus, each client separately tracks the parameters trained by its surrogate model and exits the protocol when a preset privacy budget is reached. In this paper, we determine the privacy budget based on the size of the dataset.

4 Experiment

4.1 Experimental Setup

4.1.1 Datasets

Three widely used real-world graph datasets, namely Cora, CiteSeer and PubMed, are used in this experiment. Each dataset is divided into a corresponding number of graphs according to each group of federated learning participating clients. The dataset owned by each client is part of the whole global graph. The reason for dividing the dataset by METIS is that it can split the dataset into a specified number according to the number of clients participating in federated learning, and the output of METIS can be directly used as a client dataset in non-IID scenarios. In the IID scenario, the global graph can be divided into several parts by METIS segmentation method which is far smaller than the number of clients, and then each client randomly extracts half of the nodes and related edges from the split part as the client's data set. In each setting, each client owns a corresponding dataset, and the dataset owned by each client is divided into three parts, 20% of the nodes are randomly sampled for training, 40% for validation, and 40% for testing.

4.1.2 Baselines and Our Model

1) FedAvg [3]: The most traditional standard FL algorithm, where the client sends all the learnable parameters to the server and receives the aggregated parameters from the server for the next round of training. ; 2) FedSage [1]: an advanced subgraph federated learning method that trained a three-layer GraphSage model on the FedAvg framework; 3) FedGCN [16]: This is a federated learning algorithm for graph semi-supervised node classification, which achieves excellent results in fast convergence and minimizing communication overhead. The client of FedGCN only communicates with the central server in one pre-training step, which greatly reduces communication cost, and homomorphic encryption can also be used to further enhance privacy protection. 4) Avg-push: FedAvg decentralized version and using PushSum for aggregation. 5) SD_FGL: Decentralized federated graph learning is implemented on the basis of two-channel graph neural network. The GNN network proposed before is used to train two models on each client, one for information transmission between clients, and the other for local personalized training and protecting user privacy during transmission.

4.1.3 Parameters Setting

In this experiment, three-layer GCN [22] and a linear classifier are superimposed together, where the hidden layer dimension is set to 128, the learning rate is set to 0.01, and PyTorch is used as a deep learning library. All experiments were performed on an NVIDIA Tesla A30 GPU. Since the number of rounds used in this experiment is set to 100. The similarity scale μ is set to 3 in the similarity calculation. In the experiment, all the other optimizers were optimized using Adam, and the parameters α and β of SD_FGL were set to 0.5. We set the clipping threshold C to 1 and the privacy budget ε to 1.

4.2 Experimental Results

4.2.1 Main Results

In this experiment, the three data sets are classified into corresponding non-IID and IID forms, which are utilized for node classification scenarios, as illustrated in Fig. 2 below. Fig. 3 presents the experimental results of the datasets in both non-IID and IID scenarios. Our findings indicate that SD_FGL either matches or surpasses the performance of other centralized graph federated learning models. In traditional FL, the efficiency of FedAvg is significantly diminished due to the non-independent and non-identical distribution of data among clients. Although the baseline models FedSage+ and FedGCN demonstrate improved learning efficiency in non-IID scenarios, their accuracy significantly declines when LDP is implemented. The decentralized graph federated learning scheme proposed in this paper achieves accuracy comparable to that of several recent centralized federated graph learning models and outperforms other baselines. This demonstrates that our SD_FGL effectively achieves decentralization in real-world scenarios and exhibits superior performance in federated learning tasks, thereby validating the effectiveness of the proposed framework.

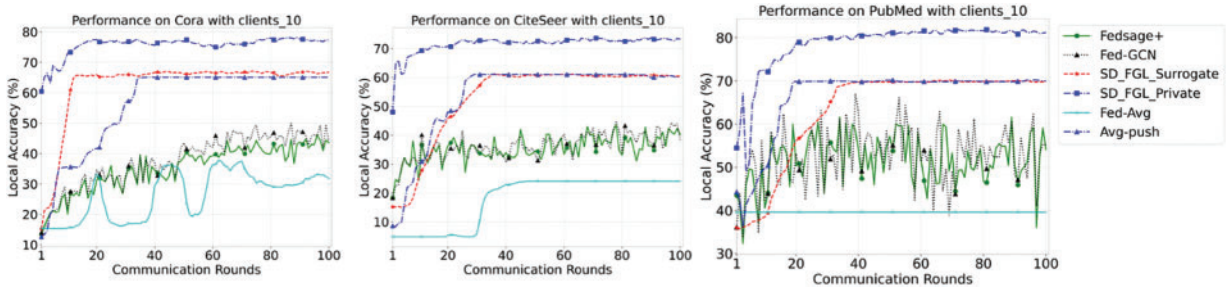

Figure 2: Convergence diagram for non-IID scenarios. 100 FL rounds with 10 clients

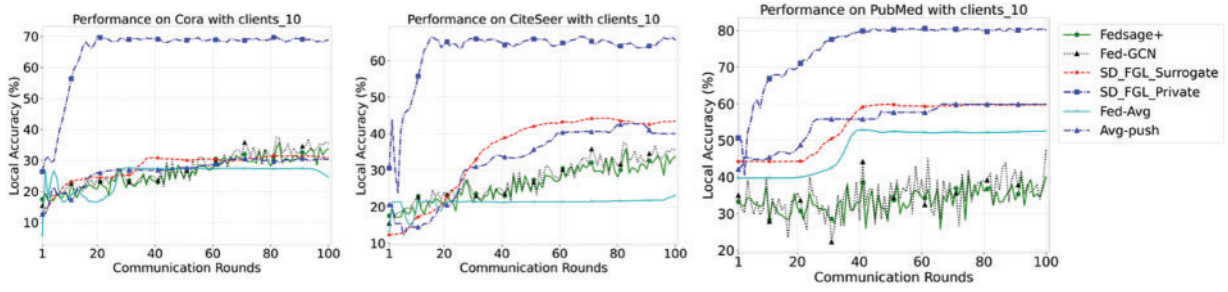

Figure 3: Convergence diagram for IID scenarios. 100 FL rounds with 10 clients

4.2.2 Convergence Analysis

It can be observed from Figs. 2 and 3 that the proposed scheme in this paper exhibits a faster convergence speed in both IID and non-IID scenarios. This is attributed to its utilization of globally shared foundational structural knowledge, which facilitates similarity calculations on features and federated learning in related domains, ultimately leading to quicker convergence. Consequently, fewer federated learning rounds are required to attain the desired test accuracy.

4.2.3 Ablation Experiments

In order to evaluate the effectiveness of decentralized federated graph learning in this experiment, an ablation study was conducted. This study involved comparing each federated learning method with and without LDP under non-IID conditions using the CiteSeer dataset. As illustrated in Fig. 4, all methods outperformed conventional training that lacked privacy constraints. Notably, the proposed framework exhibited the least performance degradation after the implementation of LDP.

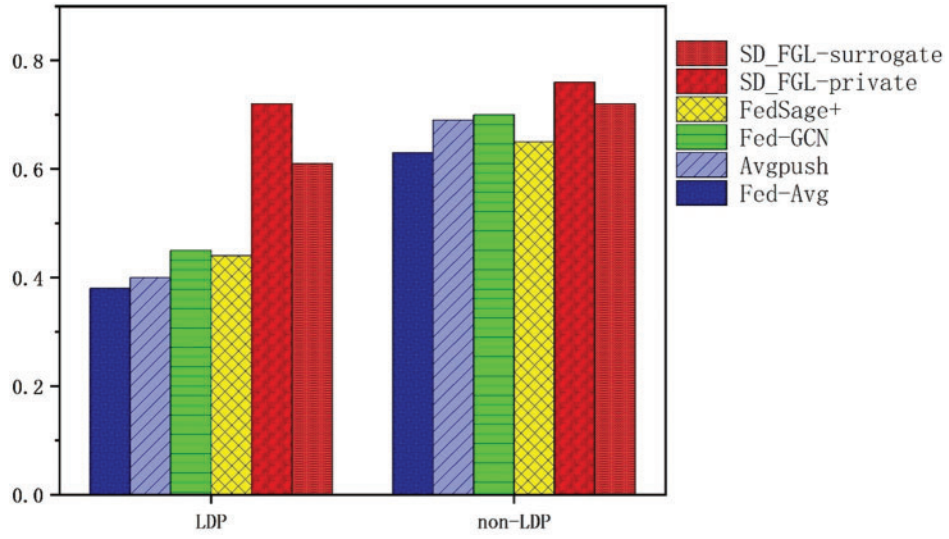

Figure 4: Accuracy changes of different models after adding local differential privacy

To evaluate the performance of the model under varying values of α and β, we observe from Fig. 5 that when $\alpha = \beta$, the performance of the private model reaches its peak, while the surrogate model demonstrates relatively stable performance. Although the accuracy gap between the surrogate model and the private model gradually narrows as α increases, the accuracy of the private model also declines.

4.2.4 Communications Efficiency

As a decentralized scheme, the proposed framework incurs lower communication costs compared to traditional centralized federated learning, as illustrated in Fig. 6. The exponential protocol maintains a constant time complexity per round, irrespective of the number of clients, which enhances the scalability of the proposed framework.

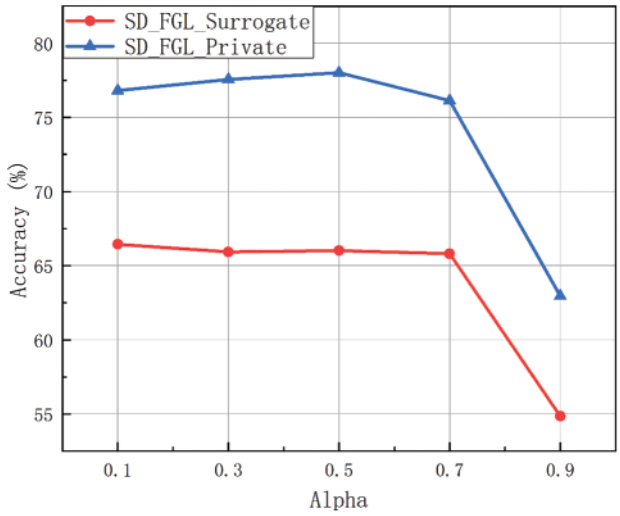

Figure 5: Accuracy of private and surrogate models under different α values

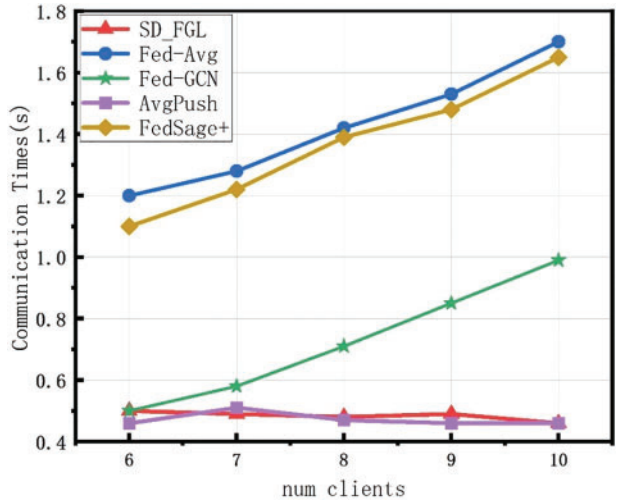

Figure 6: Comparison of communication time used by different models

5 Conclusion

Aiming to address the issues of single points of failure in central servers and high communication overhead in traditional federated graph learning, this paper proposes a decentralized Federated Graph Learning framework based on a surrogate model (SD_FGL). By maintaining both private models and surrogate models, SD_FGL not only facilitates decentralized model training but also enables the sharing and updating of surrogate models, significantly reducing communication overhead. Additionally, local differential privacy technology is incorporated to further enhance data privacy protection. Test results on three real-world graph datasets demonstrate that the SD_FGL framework achieves lower communication overhead and stronger privacy protection while improving accuracy.

Acknowledgement: The authors wish to express their appreciation to the reviewers and editors for their helpful suggestions which greatly improved the presentation of this paper.

Funding Statement: This work was supported by Inner Mongolia Natural Science Foundation Project (2021LHMS06003) and Inner Mongolia University Basic Research Business Fee Project (114).

Author Contributions: Bolin Zhang proposed Decentralized Federated Graph Learning via Surrogate Model. Bolin Zhang and Ruichun Gu conceived and designed the simulation experiment. Bolin Zhang did an experiment. Bolin Zhang and Haiying Liu analyzed the results of the simulation experiment. Bolin Zhang and Ruichun Gu wrote the manuscript text. Bolin Zhang summarized and recorded the experimental results. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data Materials: The dataset used in this study is available at https://linqs-data.soe.ucsc.edu/public/lbc/cora.tgz (accessed on 21 November 2024), https://linqs-data.soe.ucsc.edu/public/lbc/citeseer.tgz (accessed on 21 November 2024), https://linqs-data.soe.ucsc.edu/public/Pubmed-Diabetes.tgz (accessed on 21 November 2024).

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] K. Zhang, C. Yang, X. Li, L. Sun, and S. Yiu, "Subgraph federated learning with missing neighbor generation," in *35th Conf. Neural Inform. Process. Syst. (NeurIPS 2021)*, Dec. 2021, pp. 6671–6682.

[2] W. L. Hamilton, *Graph Representation Learning*. Cham: Springer, 2020. doi: 10.1007/978-3-031-01588-5.

[3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and A. B. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016, *arXiv:1602.05629*.

[4] E. T. M. Beltrán *et al.*, "Fedstellar: A platform for decentralized federated learning," *Expert. Syst. Appl.*, vol. 242, 2024, Art. no. 122861. doi: 10.1016/j.eswa.2023.122861.

[5] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen and H. Yu, "Horizontal federated learning," in *Federated Learning, Synthesis Lectures on Artificial Intelligence and Machine Learning*, Cham: Springer, 2020, pp. 49–67. doi: 10.1007/978-3-031-01585-4_4.

[6] K. Bonawitz *et al.*, "Towards federated learning at scale: System design," in *Proc. Mach. Learn. Syst.*, 2019, pp. 374–388.

[7] A. Ziller *et al.*, "PySyft: A library for easy federated learning," in *Federated Learning Systems: Towards Next-Generation AI*, 2021, pp. 111–139. doi: 10.1007/978-3-030-70604-3_5.

[8] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "BrainTorrent: A peer-to-peer environment for decentralized federated learning," 2019, *arXiv:1905.06731*.

[9] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Fully decentralized federated learning," in *Third Workshop Bayesian Deep Learn. (NeurIPS)*, 2018.

[10] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," 2019, *arXiv:1908.07782*.

[11] M. M. Amiri, D. Gunduz, S. R. Kulkarni, and H. V. Poor, "Federated learning with quantized global model updates," 2020, *arXiv:2006.10672*.

[12] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, 2020, vol. 2, pp. 429–450.

[13] M. Arivazhagan, V. K. Aggarwal, A. Singh, and S. Choudhary, "Federated learning with personalization layers," 2019, *arXiv:1912.00818*.

[14] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Netw.*, vol. 35, no. 1, pp. 234–241, 2020. doi: 10.1109/MNET.011.2000263.

[15] H. Zhang, T. Shen, F. Wu, M. Yin, H. Yang and C. Wu, "Federated graph learning–A position paper," 2021, *arXiv:2105.11099*.

[16] Y. Yao, W. Jin, S. Ravi, and C. Joe-Wong, "FedGCN: Convergence-communication tradeoffs in federated training of graph convolutional networks," in *37th Conf. Neural Inform. Process. Syst. (NeurIPS 2023)*, 2024.

[17] X. Ying, C. Liu, and D. Hu, "GCFL: Blockchain-based efficient federated learning for heterogeneous devices," in *2023 IEEE Symp. Comput. Commun. (ISCC)*, 2023, pp. 1033–1038. doi: 10.1109/ISCC58397.2023.10218066.

[18] Y. Pei *et al.*, "Decentralized federated graph neural networks," in *Int. Workshop Federated Transf. Learn. Data Sparsity Confidentiality Conjunction IJCAI*, 2021.

[19] W. Lin, B. Li, and C. Wang, "Towards private learning on decentralized graphs with local differential privacy," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 2936–2946, 2022. doi: 10.1109/TIFS.2022.3198283.

[20] C. Chen *et al.*, "Vertically federated graph neural network for privacy-preserving node classification," in *31st Int. Joint Conf. on Artificial Intell., IJCAI 2022*, 2022, pp. 1959–1965. doi: 10.24963/ijcai.2022/272.

[21] V. Dwivedi, A. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Graph neural networks with learnable structural and positional representations," 2022, *arXiv:2110.07875*.

[22] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017, *arXiv:1609.02907*.