Decentralized Federated Graph Neural Networks

Yang Pei[1⋆], **Renxin Mao**[1 *], **Yang Liu**[1], **Chaoran Chen**[2], **Shifeng Xu**[1], **Feng Qiang**

[1] Blue Elephant Tech [2] Peking University

{xuanwei.mao, yang.pei}@trustbe.cn, f.qiang@gmail.com,
chenchaoran@pku.edu.cn, bcds2018@foxmail.com

Abstract

Due to the increasing importance of user-side privacy, federated graph neural networks are proposed recently as a reliable solution for privacy-preserving data analysis by enabling the joint training of a graph neural network from graph-structured data located at different clients. However, existing federated graph neural networks are based on a centralized server to orchestrate the training process, which is unacceptable in many real-world applications such as building financial risk control models across competitive banks. In this paper, we propose a new Decentralized Federated Graph Neural Network (D-FedGNN for short) which allows multiple participants to train a graph neural network model without a centralized server. Specifically, D-FedGNN uses a decentralized parallel stochastic gradient descent algorithm DP-SGD to train the graph neural network model in a peer-to-peer network structure. To protect privacy during model aggregation, D-FedGNN introduces the *Diffie-Hellman key exchange* method to achieve secure model aggregation between clients. Both theoretical and empirical studies show that the proposed D-FedGNN model is capable of achieving competitive results compared with traditional centralized federated graph neural networks in the tasks of classification and regression, as well as significantly reducing time cost of the communication of model parameters.

1 Introduction

Graph neural networks (GNNs) have been popularly used in analyzing graph structured data, such as molecules, social, biological, and financial networks [Xu *et al.*, 2018]. However, due to user-side privacy, regulation restrictions, and commercial competition, we have observed an increasing number of cases where graph data are decentralized, which limits the applications of graph neural networks.

Recently, a few number of centralized federated graph neural networks [Mei *et al.*, 2019; Zheng *et al.*, 2021; Wu *et al.*, 2021] are proposed to enable the collaborative training of a GNN model from graph data located at different clients by using a centralized federated learning where a centralized server orchestrates the training process. For example, a recent model FedGraphNN [He *et al.*, 2021] uses FedAvg to train a centralized federated GNN model, where a client only needs to compute the embedding of self-held graph data, instead of a physically centralized dataset, to learn a local GNN model and upload the model to a central server. The central server updates the model by aggregating model parameters from all the clients, and then synchronizes the updated model to all the clients. This way, the server can build a global GNN model without any access to the raw data, and meanwhile, maintain almost the same results with the model trained directly on a centralized dataset.

Table 1: Algorithm Complexities

Algorithms	communication	computation
Centralized	$\mathcal{O}(n)$	$\mathcal{O}\left(\frac{n}{\epsilon}+\frac{1}{\epsilon^2}\right)$
Decentralized	$\mathcal{O}(Deg(network))$	$\mathcal{O}\left(\frac{n}{\epsilon}+\frac{1}{\epsilon^2}\right)$

n is the number of nodes in a network; $Deg(network)$ denotes the maximal degree of nodes, which is a constant; ϵ refers to an algorithm as ϵ-approximation to the optimal solution.

However, centralized federated GNNs face the challenge that a central server is always required to conduct model aggregation. In many cross-silo scenarios, the existence of a high-order central server is unacceptable. Taking bank fraud detection in the financial market as an example, it is often the case that there is a competitive relationship between banks participating in a federated learning task, and none of the banks accepts others to be the leader which has the full control of model updating. Therefore, a decentralized learning model is essential to real-world applications.

Another observation is that current centralized federated learning models on graph data rarely consider communication cost on the busiest central node. As shown in Table 1, existing federated graph neural networks often adopt a star network topology, and thus the central server has to deal with tens of times more communication load than clients. When a learning model has a large number of parameters, which is often the case in deep learning models, the central server will

*Equal Contribution

become the bottleneck of the learning and lower the efficiency of the entire system.

In this paper, we present a new Decentralized Federated Graph Neural Network model (D-FedGNN for short) which allows multiple participants to jointly train a graph neural network model without depending on a central node to control model updating. D-FedGNN is built upon a decentralized parallel stochastic gradient descent algorithm (DP-SGD) [Lian *et al.*, 2018] which can balance communication loads of all the participant nodes. To protect privacy during model updating, D-FedGNN introduces the Diffie-Hellman key exchange method [Bonawitz and et al., 2017] to achieve secure model aggregation between clients. Both theoretical and empirical studies have demonstrated the utility of the proposed D-FedGNN method. The contributions of this paper are summarized as follows:

- We first study the problem of decentralized federated learning on graph data that enables multiple participants to collaboratively train a graph neural network model without depending on a central server.
- We propose a new D-FedGNN model based on a decentralized parallel stochastic gradient descent algorithm (DP-SGD) and the Diffie-Hellman key exchange method to enable decentralized learning of graph neural networks with privacy protection.
- We both theoretically and empirically study the performance of D-FedGNN and the results show that D-FedGNN is competitive in terms of model accuracy and communication efficiency.

2 Related Work

In this part, we survey the related work on federated learning on graphs, decentralized federated learning, and privacy-preserving model aggregation methods.

Federated learning on graphs

Federated learning represents a new class of distributed learning models that enables model training on decentralized user data [Hegedűs *et al.*, 2019]. Recently, federated learning has been used to train graph data and several federated graph neural networks have been proposed by leveraging the power of federated learning and graph neural networks [Zheng *et al.*, 2021], [Wu *et al.*, 2021]. These models, according to their studies, perform similarly with popular centralized models in terms of model accuracy, while providing data privacy protection. In particular, the work [Mei *et al.*, 2019] uses a similarity matrix construction method to hide the private structural information of nodes. The work [Lalitha *et al.*, 2019] uses a peer-to-peer distributed federated learning framework and theoretically proves the probability of error and true risk for each node. The work [Wu *et al.*, 2021] proposed a federated graph neural network model FedGNN for recommendation systems, and introduced the differential privacy to enhance data security during model updating.

However, all these works assume that there is a central server to aggregate model information collected from client nodes. Such a strong assumption restricts further development of federated learning models on graph data in real-world applications. In contrast, we propose in this paper a new method D-FedGNN which enables a fully decentralized model for graph data with privacy protection.

Decentralized federated learning

Motivated by some cross-silo scenarios where a high-order central server is unacceptable, a number of decentralized federated learning models are proposed. Typically, the work [Lalitha *et al.*, 2018] formally described the fully decentralized federated learning problem, and presented an efficient distributed learning algorithm. The work [Hegedűs *et al.*, 2019] introduces the decentralized learning framework Gossip [Liu *et al.*, 2018] into federated learning, and proposed an alternative optimization method for federated learning. The work [Hu *et al.*, 2019] proposed a new decentralized federated learning algorithm based on both the Gossip algorithm and the model segmentation, where local models are propagated over a peer-to-peer network topology through a sum-weight gossip. The work [Liu *et al.*, 2019] designs a cross-silo federated random forest model that implements both decentralized learning and data revocation. The work [Pappas *et al.*, 2021] proposed a fully decentralized federated learning framework partially based on the interplanetary file system, which can achieve competitive performance to centralized federated learning and save both computation and communication resources. The works [Zhao *et al.*, 2019] [Li *et al.*, 2020] remove the centralized node and synchronizes federated learning updates among the data nodes, where a blockchain is used as an effective decentralized storage to replace the central sever. However, they still face the challenge of heavy communication cost.

Although the above works can fulfill decentralized federated learning, none of them considers the problem of decentralized federated learning on graph data.

Privacy-preserving model aggregation

Although federated learning can avoid data sharing during model training, it has been shown that data privacy cannot be fully guaranteed, because adversaries can still extract private information from the model parameters transmitted from a client node. A recent solution [Bonawitz and et al., 2017] is proposed based on the secure aggregation primitive at the expense of adding extra communication and computation resources. The work [Mandal and Gong, 2019] applies the secure aggregation method and homomorphic encryption to logistic regression models and linear regression models. These methods use the Diffie-Hellman key exchange method [Bonawitz and et al., 2017] to share keys between clients. The work [Geyer *et al.*, 2017] proposed the client-sided differential privacy preserving federated optimization based on differential privacy. The aim is to hide clients' contributions during training, balancing the trade-off between privacy loss and model performance. The work [Wu *et al.*, 2021] proposed to use local differential privacy federated for training graph neural networks with privacy-preserving model aggregation.

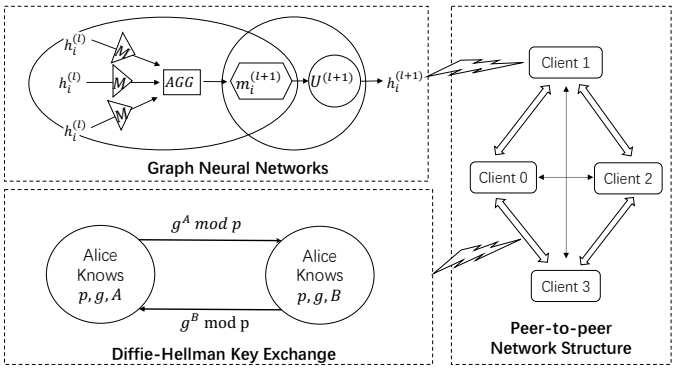

Figure 1: An illustration of the decentralized federated graph neural network D-FedGNN. D-FedGNN mainly consists of three components, i.e., a graph neural network model, a peer-to-peer network structure, and a Diffie-Hellman key exchange method.

3 Methods

In this section, we introduce the decentralized federated graph neural network model D-FedGNN in detail.

3.1 Overview

As shown in Figure 1, D-FedGNN consists of three components, i.e., a graph neural network model, a peer-to-peer network structure, and a Diffie-Hellman key exchange method.

Algorithm 1 shows the steps of D-FedGNN, there are mainly three parts of D-FedGNN, namely *system setup and initialization* (Section 3.2), *local model updating* (Section 3.3) , and *secure model aggregation* (Section 3.4). At the system setup and initialization part (Lines 1-8), D-FedGNN randomly initializes model weights, aggregation matrix, and Diffie-Hellman shared key among clients. At the local model update part (Lines 10-18), D-FedGNN calculates node embeddings by using message propagation in the graph neural network and obtains the updated model weights. At the secure model aggregation part (Lines 19-24), D-FedGNN aggregates all of the locally updated model weights based on the Diffie-Hellman key to prevent the leakage of model information. The algorithm iterates the parts of local model updating and secure model aggregation until convergence. In the following, we introduce the three parts in detail.

3.2 Setup and Initialization

The entire graph is denoted as $G = (V, E)$. A set of clients $\mathcal{P}$ are involved into the training process, each of which is attached with a unique id i and owns a private subgraph $G^i = (V^i, E^i) \subset G$. The edges E^i encode various types of relationships among nodes, such as similarities, correlations, and causal dependencies. The graph can be also described by a weighted adjacency matrix $D \in \mathcal{R}^{n\times n}$, and each client is associated with a data vector $x^i \in \mathcal{R}^d$, e.g., bag-of-words of a text or a vector of user features in a social network. Table 2 summarizes the major symbols and notations in the paper.

During the system setup and initiation, client 0 intializes trainable model weights and a decentralized weighted communication topology with an undirected graph $(\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ denotes a set of n computational nodes and $\mathcal{A} \in \mathcal{R}^{n\times n}$ is

Algorithm 1: Decentralized Federated Graph Neural Networks (D-FedGNN)

Input: Graph $G^i = (V^i, E^i)$ with node features $\{x^i, \forall i \in \mathcal{P}\}$ on data holder i; number of layers in GNN l; non-linearity function δ; adjacency matrix $D^i \in \mathcal{R}^{n\times n}, \forall i \in \mathcal{P}$; learning rate α^i of client i, aggregation weight matrix $\mathcal{A}$.
Output: the updated model weight W^i_{t+1}.

1 # Initialize and share secret key.
2 Randomly initialization $W^i_0 = \{W^i_{l,0}, \forall l \in K\}, \forall i \in \mathcal{P}$.
3 All clients synchronize the public parameter p, g and generate its own private key $kpri$.
4 **for** *client* $i \in \mathcal{P}$ **do**
5 Compute public key $kpub = g^{kpri} \mod p$.
6 Send $kpub$ to all its neighbors.
7 Receive $kpub_j$ from its neighbors j.
8 Compute the shared key $sk_{i,j} = kpub_j{}^{kpri} \mod p$.
9 **for** *each round* $t \in [0, T]$ **do**
10 # Local model update
11 **for** *client* $i \in \mathcal{P}$ **do**
12 # Calculate the initial node embeddings
13 $H^i_{1,t} \leftarrow \delta(D \cdot x^i \cdot W^i_{0,t}), \forall i \in \mathcal{P}$.
14 **for** $k \in [1, 2, ..., l-1]$ **do**
15 $H^i_{k+1,t} \leftarrow \delta(D \cdot H^i_{k,t} \cdot W^i_{k,t}))$.
16 # Get predicted label.
17 $\hat{y}^i_t \leftarrow softmax(D \cdot H^i_{K,t} \cdot W^i_{K,t}))$.
18 $W^i_{t+\frac{1}{2}} \leftarrow W^i_t - \alpha^i_t \nabla \mathcal{L}(\hat{y}^i_t, y^i_t)$.
19 # Secure model aggregation.
20 **for** $i \in P$ **do**
21 Mask model parameters $W^i_{t+\frac{1}{2}}$ based on Eq. (6) and send it to all the neighbors.
22 Receive all the masked model parameters from neighbors, recover masked model aggregation based on Eq. (7).
23 # Compute the neighborhood weighted average by fetching masked optimization variables from neighbors.
24 $W^i_{t+1} = \sum_{j\in\mathcal{P}} \left[W^j_{t+\frac{1}{2}}\right] \cdot \mathcal{A}_{i,j}$

a symmetric doubly stochastic matrix which satisfies the following three conditions: (i) $\mathcal{A}_{i,j} \in [0, 1], \forall i \in \mathcal{V}, \forall j \in \mathcal{V}$, (ii) $\mathcal{A}_{i,j} = \mathcal{A}_{j,i}$, and (iii) $\sum_j \mathcal{A}_{i,j} = 1$. $\mathcal{A}_{i,j}$ represents the weight between clients i and j. $\mathcal{A}_{i,j} = 0$ means that the two clients are disconnected. Take the standard ring network topology as an example. $\mathcal{A}$ can be represented as follows,

$$\mathcal{A} = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & & & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & & \\ & \frac{1}{3} & \frac{1}{3} & \ddots & \\ & & \ddots & \ddots & \frac{1}{3} \\ \frac{1}{3} & & & \frac{1}{3} & \frac{1}{3} \end{pmatrix} \in \mathcal{R}^{n\times n}. \quad (1)$$

Then, client 0 sends trainable model parameters and decentralizedd weighted communication topology to other client. At Last, each client i invokes the Diffie-Hellman key exchange protocol to establish a shared key $sk_{i,j}$ with client

Table 2: Symbols and Notations.

Notations	Descriptions
$\mathcal{P}$	union set of clients
G^i	graph data at client i
V^i	nodes of graph G^i
E^i	edges of graph G^i
x^i	features of graph G^i
D^i	adjacency matrix of graph G^i
y_t^i	labels of graph G^i
$\hat{y}_t^i$	predicted labels of of graph G^i
J	the total number of classes
$y_{t,j}^i$	label probability of class j at epoch t
$\hat{y}_{t,j}^i$	predicted label probability of class j at epoch t
α^i	learning rate for graph G^i at client i
W_t^i	model weight of client i at the t-th epoch
$[W_i]$	masked model weights
$\mathcal{A}$	weighted matrix for model aggregation
X	node embedding vertor as the GNN input

j. Note that for any $i \neq j$, we have $sk_{i,j} = sk_{j,i}$.

3.3 Local Model Updating

Each client locally trains a graph neural network model with respect to its own private data $G^i = (V_i, E_i)$. Without loss of generality, we use GCNs as an example. During the training process, each client i first derives node embedding vectors X as introduced in the work [Kipf and Welling, 2016]. With the node embedding X, the client runs a forward computation of a graph convolutional layer as follows:

$$H = \delta(D \cdot X \cdot W), \tag{2}$$

where $delta$ is an element-wise non-linear function (e.g., a ReLU function). The adjacency matrix D encodes edge information in the graph data, which can be a graph shift operator such as the Laplacian matrix. Consider that there are l layers in total for graph convolution, the output of an GCN can be denoted as follows,

$$\hat{y}_t^i = f(D, W; H_l(X)) = softmax(D \cdot H_l(X) \cdot W), \tag{3}$$

where a softmax function is used to get the normalized probability. Then, the loss function $\mathcal{L}(\cdot)$ can be defined as follows,

$$\mathcal{L}(\hat{y}_t^i, y_t^i) = -\frac{1}{n^i} \sum_{k=1}^{N} \sum_{j=1}^{J} y_{t,j}^i \log \hat{y}_{t,j}^i. \tag{4}$$

Based on the above output and loss function, the backward computation is conducted to update local model as follows,

$$W_t^i \leftarrow W_{t-1}^i - \alpha^i \nabla \mathcal{L}(\hat{y}_t^i, y_t^i), \tag{5}$$

where t specifies the current number of iterations, α^i is the learning rate, y_t^i is the true label owned by client i, and $\nabla \mathcal{L}(\hat{y}_t^i, y_t^i)$ is the gradient.

3.4 Secure Model Aggregation

Each client securely aggregates the newly updated model parameters with its neighbors. Specifically, a client i first securely masks its local model parameters W_i as follows,

$$[W_i] = W_i + \sum_{i<j} PRG(sk_{i,j}) - \sum_{i>j} PRG(sk_{i,j}), \tag{6}$$

where $PRG(\cdot)$ is a pseudo-random number generator [Bonawitz and et al., 2017], and $sk_{i,j}$ is the shared key between clients i and j.

Then, the masked model parameter $[W_i]$ is sent to the neighbors of client i. After receiving all the masked model parameters from neighbors, the client runs model aggregation locally as follows,

$$W_{i,t+1} = \sum_{j \in \mathcal{P}} [W_j] \cdot \mathcal{A}_{i,j}, \tag{7}$$

where $W_{i,t+1}$ is exactly the final updated model parameters at iteration t. Repeat the above steps, the clients can collaboratively train a global graph neural network model without the assistance of a centralized server.

4 Analysis

In this part, we study the performance of D-FedGNN from the perspective of *generalization error rate*. Intuitively, we can estimate the generalization error bounds of D-FedGNN by combining the bound of D-SGD generalization error and the bound of graph neural networks. A recent work [Sun *et al.*, 2021] has studies the generalization error bounds of D-SGD under convex learning functions, where the uniform stability ϵ_{stab} is used to bound the generalization error which depends on the learning function, the learning rate, and the structure of the decentralized graph. On the other hand, according to a previous theoretical study on graph neural networks [Xu *et al.*, 2018], a message passing function between neighboring nodes can be taken a multi-layer neural network and the work [Cao and Gu, 2020] has studied an algorithm-dependent generalization error bound for multi-layer neural networks with ReLu. Thus, we can estimate the generalization error bound of D-FedGNN by combining their results.

Formally, let G^i be the graph located at the i^{th} client, and the objective function $F(x) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} F(x; G^i)$, where n is the average size of each graph G^i. Basically, the gradient of $F(x; G^i)$ is L-Lipschitz. Let $(x^t)_{1 \leq t \leq T}$ be a sequence generated by D-FedGNN, with β_t be the step size at round t, and the average of all the T steps be $ave(x^T) = (\sum_{t=1}^{T-1} \beta_t x^t) \backslash (\sum_{t=1}^{T-1} \beta_t)$. Regarding the structure of the connected graph, let $\mathcal{A}$ be a mixing matrix [Marshall *et al.*, 1979] defined on the graph edges which is a doubly stochastic matrix, and use a constant λ to denote matrix $\mathcal{A}$, i.e. $\lambda = \max\{|\lambda_2|, |\lambda_m(\mathcal{A})|\}$, where λ_i denotes the i^{th} largest eigenvalue of $\mathcal{A}$. Regarding the multi-layer neural networks used in graph neural networks, we use $\mathcal{E}_G(W)$ to denote the empirical surrogate error on a training graph G. For any given parameters $\eta > 0$ and a large enough absolute constant $C \geq \tau \backslash (L^6 \log(m)^{3 \backslash 2})$, the generalization error ϵ_{gen} is bounded by the uniform stability [Hardt *et al.*, 2016] on the average of all the T steps $avg(x^T)$ as follows:

Theorem 1. *(Generalization Bound ϵ_{gen}) Let the step size $\beta_t \equiv 2 \backslash L$, then uniform stability $ave(x^T)$ with the probability at least of $1 - \eta$ to satisfy:*

$$\begin{aligned} \epsilon_{gen} \leq & \frac{1}{mn} 2C\sqrt{m}\mathcal{E}^2 \beta(t-1) \\ & + \frac{1_{\lambda \neq 1}}{1-\lambda} 4\beta C\sqrt{m}\mathcal{E}^2(1 + \beta C\sqrt{m}\mathcal{E})(t-1), \end{aligned} \tag{8}$$

Proof. Because D-FedGNN is built on D-SGD and graph neural networks, the generalization bound is based on both. According to the work [Sun *et al.*, 2021], the generalization bound of D-SGD with respect to the average steps $ave(x^T)$ depends on the maximal gradient of the learning function ∇F, the learning rate β, and the structure of the connected graph λ as: $\epsilon_{gen} \leq \frac{2\nabla F^2 \beta(t-1)}{mn} + \frac{4\beta\nabla F^2(1+\beta\nabla F)(t-1)}{1-\lambda} 1_{\lambda \neq 1}$. On the other hand, the message passing in graph neural networks is a multi-layer neural network with ReLu which satisfies the bound of $\max \|\nabla_w F\| \leq C\sqrt{m\mathcal{E}}$ with the probability of $1-\eta$ for an absolute constant $C \geq \tau \backslash (L^6 \log(m)^{3 \backslash 2})$ [Cao and Gu, 2020]. Thus, by plugging the gradient $\nabla_W F$ into D-SGD, we obtain Eq.(8). □

Theorem 1 shows that the generalization capability of D-FedGNN depend on the empirical surrogate error $\mathcal{E}$ based on the learning function ∇F and training graph data, the learning rate β, and the structure of the decentralized network λ. In addition, based on the work [Sun *et al.*, 2021], for a stochastic algorithm on a dataset, we can also use the excess generalization error to bound the generalization error which can be decomposed into three parts, i.e. a generation error, an optimization error, and a test error. Due to space, we omit the discussion.

5 Experiments

In this section, we conduct experiments to validate whether D-FedGNN performs competitively compared with existing centralized graph neural networks, and whether D-FedGNN takes less time during model parameters communication.

5.1 Experimental settings

All the experiments were run on a GPU server with 4 NVIDIA GEFORCE 2080Ti (11GB GPU memory).

Datasets

The benchmark datasets are summarized in Table 3. Moreover, we use an imbalanced partition algorithm Latent Dirichlet Allocation (LDA) [He *et al.*, 2020] to partition datasets in the MoleculeNet benchmark [Wu *et al.*, 2018]. The value of α for LDA at each non-I.I.D. graph dataset is shown Tables 4 and 5. Also, we use random splitting as advised in the work [Wu *et al.*, 2018] to partition the datasets into three parts, e.g., 80% for training, 10% for validation, and 10% for testing.

Parameter settings

According to the recent work [He *et al.*, 2021], we tune hyper-parameters by using grid search. The hyper-parameters include the learning rate $\alpha \in [0.0015, 0.015, 0.15]$, dropout rate $d \in [0.3, 0.5, 0.6]$, and rounds of iterations $r \in [10, 50, 100]$. All the hyper-parameters are tuned on a single GPU. The mini-batch size is fixed to be 1 as required by the standard molecule tasks [Wu *et al.*, 2018]. Besides, we set the dimension of node embedding, hidden layer, readout embedding and graph embedding to be $(64, 64, 64, 64)$.

Metrics

By following the work [He *et al.*, 2021], we use ROC-AUC as the evaluation metric for classification tasks. We use R-MSE as the evaluation metric for regression tasks. In decentralized scenarios, we use the average results collected from all the clients. To compare the communication efficiency, we record the run times of the models as well.

Table 3: Summary of datasets

Dataset	Compounds	Average of Nodes	Average of Edges
ESQL	1128	13.29	40.65
FreeSolv	642	8.72	25.60
Lipophilicity	4200	27.04	86.04
hERG	10572	29.39	94.09
BACE	1513	34.09	36.89
BBBP	2039	24.03	25.94
SIDER	1427	33.64	35.36
ClinTox	1478	26.13	27.86
Tox21	7831	18.51	25.94

Benchmark methods

We compare the proposed D-FedGNN method with a recent centralized GNN model FedGraphNN [He *et al.*, 2021] [1]. FedGraphNN is an open-sourced federated learning framework for GNNs, which has implemented the standard baseline datasets, models, and federated learning algorithms for GNNs. Note that FedGraphNN is based on FedAVG which utilizes a central server to aggregate client model parameters and maintain the global model.

For both D-FedGNN and FedGraphNN, we implement three popular GNNs to train local models, i.e., GCN, GAT and GraphSAGE. A readout function (a simple multilayer perceptron) is used to handle the output of the GNN models.

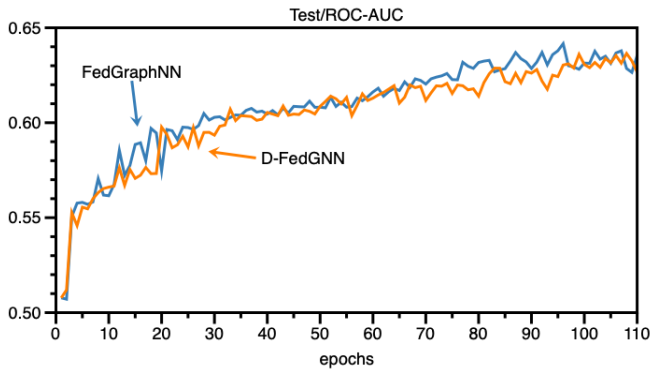

Figure 2: The ROC-AUC results of GCN on sider *w.r.t.* epochs.

5.2 Experimental results

In Table 4 and Table 5, we list the results of regression with respect to RMSE and the results of classification with respect to ROC-AUC respectively.

From the results, we can observe that D-FedGNN performs similarly with the centralized model FedGraphNN on

[1] https://github.com/FedML-AI/FedGraphNN

Table 4: Regression results *w.r.t.* RMSE on 4 clients.

Datasets	Non-I.I.D Partition	GNNs	FedGraphNN	D-FedGNN
FreeSolv (642)	LDA $\alpha = 0.5$	GCN	2.7470	2.6402
		GAT	1.3130	1.3523
		GraphSAGE	1.6410	1.6734
ESOL (1128)	LDA $\alpha = 0.5$	GCN	1.4350	1.4552
		GAT	0.9643	0.9404
		GraphSAGE	0.8604	0.8534
Lipo (4200)	LDA $\alpha = 0.5$	GCN	1.1460	1.2452
		GAT	0.8537	0.8423
		GraphSAGE	0.7788	0.7553
hERG (10572)	LDA $\alpha = 2$	GCN	0.7944	0.7743
		GAT	0.7322	0.7497
		GraphSAGE	0.7265	0.7563

Table 5: Classification results *w.r.t.* ROC-AUC on 4 clients.

Datasets	Non-I.I.D Partition	GNNs	FedGraphNN	D-FedGNN
SIDER (1427)	LDA $\alpha = 0.2$	GCN	0.638	0.626
		GAT	0.6591	0.6494
		GraphSAGE	0.6700	0.6734
BACE (1513)	LDA $\alpha = 0.5$	GCN	0.8784	0.6523
		GAT	0.7714	0.8604
		GraphSAGE	0.7812	0.8734
Clintox (1478)	LDA $\alpha = 0.5$	GCN	0.876	0.8933
		GAT	0.9129	0.9125
		GraphSAGE	0.9246	0.9256
BBBP (2039)	LDA $\alpha = 2$	GCN	0.7605	0.7909
		GAT	0.8746	0.8863
		GraphSAGE	0.8935	0.908
Tox21 (7831)	LDA $\alpha = 3$	GCN	0.7425	0.7256
		GAT	0.7186	0.7102
		GraphSAGE	0.7801	0.7912

Table 6: Training time (Hardware: 4 NVIDIA GEFORCE RTX 2080Ti GPU(11GB/GPU))

	GNNs	SIDER	BACE	Clintox	BBBP	Tox21
FedAVG	GCN	9.05min	9.53 min	9.03 min	11.57 min	39.16 min
D-FedGNN		7.87 min	8.32 min	8 min	10.62 min	38.05 min
FedAVG	GAT	13.72 min	13.68 min	13.06 min	16.39 min	44.45 min
D-FedGNN		12.05 min	12.65 min	8 min	10.71 min	38.09 min
FedAVG	GraphSAGE	4.43 min	8.03 min	8.47 min	9.34 min	19.20 min
D-FedGNN		3.28 min	7.12 min	7.12 min	8.18 min	18.09 min

the benchmark datasets. For example, in Table 4, D-FedGNN obtains better RMSE results on the FreeSolv dataset when using GAT and GraphSAGE, but slightly worse than FedGraphNN when using GCN. In Table 5, D-FedGNN obtains better ROC-AUC results than FedGraphNN on the BBBP dataset under all of the three GNN models. Moreover, Figure 2 and Figure 3 show two training curves with respect to ROC-AUC on two datasets, e..g, sider and clintox. From the results, we can conclude that D-fedGNN converges almost the same as FedGrapgNN.

In Table 6, we record the training times on the benchmark datasets. We can observe that D-FedGNN takes less time than FedGraphNN to run 110 federated model aggregation. This is because D-FedGNN has more balanced workload among clients than FedGraphNN. Moreover, Figure 4 gives the specific results on the clintox dataset over 500 times of model aggregation. The results show that D-FedGNN outperforms FedGraphNN with a large margin. In our experiments, we only use 4 GPUs. When the number of GPUs grows and the model size increases, the running time cost will reduce significantly.

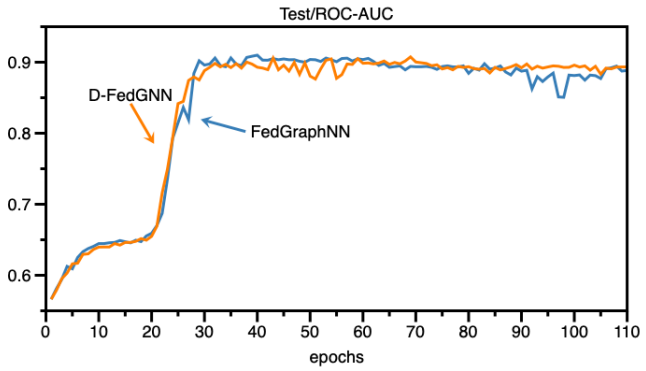

Figure 3: The ROC-AUC results of GCN on clintox *w.r.t.* epochs.

Last but not least, we list the search results of the hyper-parameters with respect to the three GNN models for readers to repeat the results. For GCNs, the learning rate is 0.0015 and the dropout rate is 0.3 for all the benchmark datasets. For GATs, the learning rate is 0.0015, dropout rate is 0.3, the attention head is 2, and α is 0.2. For GraphSAGE, the learning rate is 0.015 for the datasets BBBP, ClinTox, FreeSolv, the learning rate is 0.0015 for datasets BACE, SIDER, ESOL, Lipophilicity, hERG and the learning rate is 0.00015 for the dataset Tox21. The dropout rate is 0.6 for all the datasets. As for the Diffie-Hellman key exchange method, p specifies a 2048-bits prime field $\mathbb{Z}_p$, and g is a randomly selected generator of $\mathbb{Z}_p$.

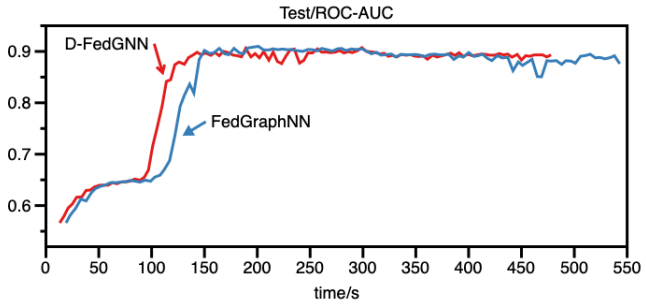

Figure 4: The ROC-AUC results of GCN on clintox *w.r.t.* time.

6 Conclusions

In this paper, we present a new Decentralized Federated Graph Neural Network model (D-FedGNN for short) to enable multiple participants to train a graph neural network model without a centralized server. D-FedGNN uses a peer-to-peer network structure and discards the central server during model updating. Moreover, to protect privacy during model updating, D-FedGNN introduces the Diffie-Hellman key exchange method [Bonawitz and et al., 2017] to achieve secure model aggregation between clients. Both theoretical and empirical studies demonstrate the utility of the proposed D-FedGNN method.

References

[Bonawitz and et al., 2017] Keith Bonawitz and et al. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[Cao and Gu, 2020] Yuan Cao and Quanquan Gu. Generalization error bounds of gradient descent for learning over-parameterized deep relu networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3349–3356, 2020.

[Geyer *et al.*, 2017] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[Hardt *et al.*, 2016] Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 1225–1234. PMLR, 2016.

[He *et al.*, 2020] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, et al. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020.

[He *et al.*, 2021] Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Yu Rong, Peilin Zhao, Junzhou Huang, Murali Annavaram, and Salman Avestimehr. Fedgraphnn: A federated learning system and benchmark for graph neural networks. *arXiv preprint arXiv:2104.07145*, 2021.

[Hegedűs *et al.*, 2019] István Hegedűs, Gábor Danner, and Márk Jelasity. Gossip learning as a decentralized alternative to federated learning. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 74–90. Springer, 2019.

[Hu *et al.*, 2019] Chenghao Hu, Jingyan Jiang, and Zhi Wang. Decentralized federated learning: a segmented gossip approach. *arXiv preprint arXiv:1908.07782*, 2019.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Lalitha *et al.*, 2018] Anusha Lalitha, Shubhanshu Shekhar, Tara Javidi, and Farinaz Koushanfar. Fully decentralized federated learning. In *Third workshop on Bayesian Deep Learning (NeurIPS)*, 2018.

[Lalitha *et al.*, 2019] Anusha Lalitha, Osman Cihan Kilinc, Tara Javidi, and Farinaz Koushanfar. Peer-to-peer federated learning on graphs. *arXiv preprint:1901.11173*, 2019.

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

[Lian *et al.*, 2018] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in Neural Information Processing Systems 30*, 8:5331–5341, 2018.

[Liu *et al.*, 2018] Yang Liu, Ji Liu, and Tamer Basar. Gossip gradient descent. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1995–1997, 2018.

[Liu *et al.*, 2019] Yang Liu, Zhuo Ma, Ximeng Liu, Zhuzhu Wang, Siqi Ma, and Ken Ren. Revocable federated learning: A benchmark of federated forest. *arXiv preprint arXiv:1911.03242*, 2019.

[Mandal and Gong, 2019] Kalikinkar Mandal and Guang Gong. Privfl: Practical privacy-preserving federated regressions on high-dimensional data over mobile networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pages 57–68, 2019.

[Marshall *et al.*, 1979] Albert W Marshall, Ingram Olkin, and Barry C Arnold. *Inequalities: theory of majorization and its applications*, volume 143. Springer, 1979.

[Mei *et al.*, 2019] Guangxu Mei, Ziyu Guo, Shijun Liu, and Li Pan. Sgnn: A graph neural network based federated learning approach by hiding structure. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2560–2568. IEEE, 2019.

[Pappas *et al.*, 2021] Christodoulos Pappas, Dimitris Chatzopoulos, Spyros Lalis, and Manolis Vavalis. Ipls: A framework for decentralized federated learning. *arXiv preprint arXiv:2101.01901*, 2021.

[Sun *et al.*, 2021] Tao Sun, Dongsheng Li, and Bao Wang. Stability and generalization of the decentralized stochastic gradient descent. *arXiv preprint arXiv:2102.01302*, 2021.

[Wu *et al.*, 2018] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[Wu *et al.*, 2021] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925*, 2021.

[Xu *et al.*, 2018] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[Zhao *et al.*, 2019] Yang Zhao, Jun Zhao, Linshan Jiang, Rui Tan, and Dusit Niyato. Mobile edge computing, blockchain and reputation-based crowdsourcing iot federated learning: A secure, decentralized and privacy-preserving system. *arXiv preprint arXiv:1906.10893*, 2019.

[Zheng *et al.*, 2021] Longfei Zheng, Jun Zhou, Chaochao Chen, Bingzhe Wu, Li Wang, and Benyu Zhang. Asfgnn: Automated separated-federated graph neural network. *Peer-to-Peer Networking and Applications*, 14(3):1692–1704, 2021.