FedSGC: Federated Simple Graph Convolution for Node Classification

Tsz-Him Cheung , **Weihang Dai** and **Shuhan Li**
The Hong Kong University of Science and Technology
{thcheungae, wdaiaj, slidm}@connect.ust.hk

Abstract

Graph Neural Networks (GNN) have developed rapidly and solved a wide range of graph-related tasks. One advantage of GNN over traditional neural networks is the utilization of neighbouring information. However, under the increasing concern in data privacy, accessing raw information from different parties may raise privacy concerns. To address this, federated learning, as a distributed learning mechanism, is proposed to train models with decentralized data owned by different data parties without sharing or leaking the raw data. In this work, we study the vertical and horizontal settings for federated learning on graph data. We propose FedSGC to train the Simple Graph Convolution model under three data split scenarios. We also demonstrate that the prediction performance of FedSGC is closely aligned with the non-federated model trained in centralized manner.

1 Introduction

Graph neural networks (GNNs) have developed rapidly in the past few years. GNN can be used to solve a wide range of graph-related tasks such as node classification, graph classification, graph generation, node clustering and link prediction [Zhang and Chen, 2018; Wang *et al.*, 2017]. Recently, there is a increasing trend of extending GNN to other machine learning tasks, including computer vision, natural language processing, traffic forecasting, recommendation systems and protein folding problem [Li *et al.*, 2017; Ying *et al.*, 2018; Fout, 2017]. One of the key success factors for GNN is the utilization of neighbour information. But recently, the society is increasingly concerned with the use of personal data. To address this, federated learning (FL) was proposed to collaboratively train a machine learning model with decentralized data without scarifying data privacy. For graph data, if the node feature is owned by individual node parties, for example user data in a social network application, accessing the neighbouring information may raise concern about data security and privacy. Therefore, there is a need to develop graph-related federated learning methods to take advantage of the recent advancement in GNN.

According to [Yang *et al.*, 2019], federated learning can be categorized into Vertical Federated Learning (VFL), Horizontal Federated Learning (HFL) and Transfer Federated Learning (TFL). In this work, we further explore the settings of VFL and HFL on graph data. In VFL, the data is split among the feature space with shared common sample-ID. The federated parties work collaboratively to train a model without revealing their feature and label information to other parties. For graph problems, in addition to the node features, we treat the topology as another form of "feature" that can be sensitive and owned by different parties. This causes more variations in the split of data, for example the topology information is hold by one party and the feature information is hold by another party. This renders the VFL on graph data a more complicated problem than the standard VFL. A real-life application scenario can be: consider two different companies, one is a social network company and the other is a e-commerce company. The social network company knows the interactions between their users, which can be represented as a form of graph data, and the e-commerce company knows the purchase preference of their users, which can be represented as a form of tabular data. If the social network company wants to perform advertisement prediction on their users, the purchase history owned by the e-commerce company is likely to be useful. However, such purchase data can be sensitive. Federated learning over the graph data and purchase data will allow the two parties to collaborate without privacy concerns.

In HFL, the data is split in the sample-ID space with shared common features. The first proposed and perhaps most widely-used Federated Averaging (FedAvg) algorithm solves HFL by averaging the models that had been trained locally, using a centralized server [McMahan *et al.*, 2017]. However, in graph problems, the learning of node representation typically requires information from other neighbouring node parties and thereby prohibiting the training of local models individually. This makes HFL on graph data more challenging than on the common tabular data, which does not require communications between horizontal participants.

In this work, we formally define three different data split scenarios for graph data and propose FedSGC to apply federated learning techniques in node classification problems. At a high level, FedSGC utilizes the linear computations in Simple Graph Convolution (SGC) [Wu *et al.*, 2019] and applies Homomorphic Encryption (HE) technique to protect the

data privacy during the exchange of information and update of model. Our contribution can be summarized as follow:

- We propose three problems of privacy-preserving learning on horizontally partitioned and vertically partitioned graph data in the setting of federated learning.
- We present a novel federated graph neural network FedSGC, which integrates Simple Graph Convolution with federated learning.
- We demonstrate that our approach performs similarly with non-federated version of SGC which is trained in a centralized manner on three citation networks datasets.

2 Preliminaries and Related Work

2.1 Federated learning

Federated learning aims to train a machine learning model while protecting the privacy of data, which is distributed across multiple parties [Konečnỳ *et al.*, 2016a; Konečnỳ *et al.*, 2016b; McMahan *et al.*, 2016]. Attempts have been made to solve the federated learning problem under different settings.

VFL For vertical federated learning setting, several previous works leverage Homomorphic Encryption (HE) to secure the data exchange. The encryption scheme allows computation over encrypted ciphertext while keeping the computed results as encrypted and match with the results computed in plaintext [Rivest *et al.*, 1978; Paillier, 1999; Acar *et al.*, 2018]. Among different levels of homomorphic encrpytion (e.g. fully homomorphic encryption and partial homorphic encryption), additively homomorphic encryption (AHE) is commonly adopted. It allows certain additive operations to be carried out between ciphertexts while maintaining certain degree of computation efficiency. Specifically, AHE requires:

$$[\![u]\!] + [\![v]\!] = [\![u + v]\!] \tag{1}$$

$$a \cdot [\![u]\!] = [\![a \cdot u]\!] \tag{2}$$

, where u, v and a are three numbers in plaintext, $[\![\cdot]\!]$ denotes the ciphertext of a number. By using AHE, one party can encrypt the intermediate results and another party can carry out addictive operations over the encrypted data to update a global model without knowing the actual data. The technique has been applied to linear models, like secure linear regression [Gascón *et al.*, 2016] and logistic regression [Hardy *et al.*, 2017]. For tree-boosting model, [Cheng *et al.*, 2019] proposed SecureBoost in VFL setting to learn a federated XGBoost model through carefully analysing the intermediate results and applying AHE to secure the communications. These methods usually achieve same prediction accuracy when compared to the non-federated version, and are simple and efficient to implement. Following their success, we analyse the operation of Simple Graph Convolution and apply HE to secure the intermediate data communications and uses addictive operations to update the model.

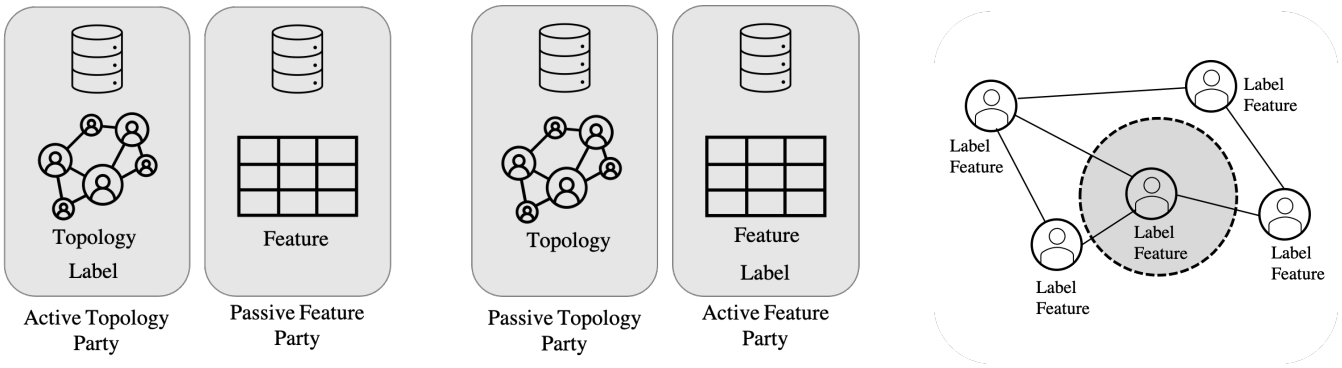
HFL For horizontal federated learning settings, multiple works have proposed that each party uses its local data to train a local model with the same architecture and later update the global model by averaging all the local models [McMahan *et al.*, 2016]. This technique is usually referred to as the Federated Averaging (FedAvg) algorithm. Although FedAvg is simple to implement, the technique cannot be extended to graph federated problem directly as in graph neural networks, the feature propagation process requires accessing the information from some node neighbours, thereby requiring an additional secure communication channel between node parties.

2.2 Federated Graph Neural Network

Recently, a few approaches have been proposed to apply federated learning to graph neural networks. For vertical federated learning on graph data, VFGNN is proposed to tackle one scenario of graph vertical federated learning, where the nodes are the same across different data parties but with different features and edges [Zhou *et al.*, 2020]. VFGNN uses relatively complex secret sharing and differential privacy techniques and does not discuss other ways to split the data. In our work, we discuss two other possible ways to split the data and propose a more efficient way to train a federated graph neural network that performs similarly with the non-federated model.

For horizontal federated learning, [Zheng *et al.*, 2021] focuses on solving the non-IID data problem among different data parties and proposes a two-stage process to train a federated graph neural network and tune the hyperparameters using Bayesian optimization. In their work, the data is partitioned horizontally: the feature of the training nodes are the same, however, the training nodes and the graphs are different among the data parties. [Wu *et al.*, 2021] considers a privacy-preserving recommendation task under horizontal federated learning setting. Depending on the iterations between the users and items, each user can form an individual user-item graph for local model training. Once a while, the local models are aggregated as the global model. In most of the proposed horizontal federated learning settings over graph data, each data party owns a private graph with the private node feature information. Each data party can hence train its local graph model and aggregate afterwards. In our work, we consider a different scenario, where each party is a single node with feature and connections to its immediate neighbours. Notice that this setting is slightly more complicated but is commonly found in real-life situations. For example, each user account in a social network should only know its private information and the neighbours (friends) it is connected to. Unlike the previous settings, each node can no longer train its local model individually since the training of graph neural network typically requires the aggregation of feature information among different node parties. Securing such feature information in model training is a challenging problem to address.

Compared to the application of federated learning in other data modalities, such as tabular data, image data and text data, there is less work focusing on graph data. To the best of our understanding, there is no other literature considering the same data split as we proposed. Moreover, existing methods are not designed in a communication efficient way and are hard for practical uses.



(a) Case 1 (b) Case 2 (c) Case 3

Figure 1: Illustration of three types of data partitions in federated node classification.

3 Method

3.1 Problem Setting

We propose three data split scenarios for federated node classification task:

- Case 1: Two parties collaborate to train a SGC model in a vertical federated learning setting. The first party owns the topology and label information (*Active Topology Party*). The second party owns the feature information (*Passive Feature Party*).
- Case 2: Two parties collaborate to train a SGC model in a vertical federated learning setting. The first party owns the topology information (*Passive Topology Party*). The second party owns the feature and label information (*Active Feature Party*)
- Case 3: Multiple node parties collaborate to train a SGC model in a horizontal federated learning setting. Each node party owns its feature, the connections to its 1-hop neighbour and its label.

The three cases are illustrated in Figure 1. In the following sections, we first review and analysis the Simple Graph Convolution (SGC) model, then explain three FedSGC workflows to train the SGC model for the three proposed cases.

3.2 Overview of Simple Graph Convolution (SGC)

In graph convolution, there are two major processes: feature propagation and feature transformation. At each layer of the graph convolution, the feature of each node is aggregated with the features from its neighbouring nodes. Then, the aggregated feature is transformed linearly. Canonical Graph Convolution Network (GCN) further applies a nonlinear activation function, typically as ReLU, and outputs the result as the node feature of a specific layer [Kipf and Welling, 2017]. To reduce the computation effort, [Wu *et al.*, 2019] proposed SGC, which linearized the GCN operations, and found that it performs comparably with other baselines in several node classification tasks. SGC removes the nonlinear activation functions in GCN and computes the prediction with S denotes the normalized adjacency matrix with self-loop, Θ^k denotes the learnable linear transformation weight matrix at layer k and σ is the final nonlinear activation function at the last layer for classification task:

$$S = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} \tag{3}$$

$$\hat{Y} = \sigma(S...SX\Theta^{(1)}...\Theta^{(K)}) \tag{4}$$

where $\tilde{A} = A + I$ is the adjacency matrix with self-loop and $\tilde{D}$ is the diagonal degree matrix of $\tilde{A}$. By multiplying S for K times, each node can aggregate the information up to its K-hop neighbourhood. The expression can be further simplified as:

$$\hat{Y} = \sigma(S^K X\Theta) \tag{5}$$

Equation 5 can be viewed as a logistic regression problem with σ as the sigmoid function and $S^K X$ as the "preprocessed" feature set. Such formulation shows certain linear relationships between the topology, the feature and the parameters, which favours the use of addictively homomorphic encryption to protect the data privacy in federated learning setting. Here, we further state the gradient of the binary cross entropy loss w.r.t the the SGC parameters:

$$\hat{Y} = \sigma(S^K X\Theta) = \frac{1}{1 + e^{-S^K X\Theta}} \tag{6}$$

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} Y_i \log \hat{Y}_i + (1 - Y_i)\log(1 - \hat{Y}_i) \tag{7}$$

$$\frac{\partial \mathcal{L}}{\partial \Theta} = \frac{1}{n}(\hat{Y} - Y)^T (S^K X) \tag{8}$$

3.3 FedSGC

Case 1: FedSGC with Vertical Active Topology party (FedSGC-VAT) First, we consider a federated learning setting with two parties. The first party (party A) owns the topology information S and target label Y, where the second party (party B) owns the feature X and the parameter Θ (Figure 1a). To train a federated SGC, we need to protect the normalized adjacency matrix S, Y and raw feature X not to be learned by other parties during the model training. Here, we

Step	Active Topology Party A (with S, Y)	Passive Feature Party B (with X)
0	Create encryption key pairs and send the public key to B, initialize K	Initialize Θ
1		Compute and send $X\Theta$ to A
2	Compute $\hat{Y} = \sigma(S^K X\Theta)$ and send $[\![\frac{1}{n}(\hat{Y} - Y)^T S^K]\!]$ to B	
3		Compute $[\![\frac{\partial \mathcal{L}}{\partial \Theta}]\!] = [\![\frac{1}{n}(\hat{Y} - Y)^T S^K]\!]X$ and send $[\![\frac{\partial \mathcal{L}}{\partial \Theta}]\!] + [\![R_B]\!]$ to A
4	Decrypt $[\![\frac{\partial \mathcal{L}}{\partial \Theta}]\!] + [\![R_B]\!]$ and send $\frac{\partial \mathcal{L}}{\partial \Theta} + R_B$ to B	
5		Update Θ

Table 1: Model training procedure of FedSGC-VAT

Step	Passive Topology Party A (with S)	Active Feature Party B (with X, Y)
0	Create encryption key pairs and send the public key to B, initialize K	Initialize Θ
1	Compute S^K and send $[\![S^K]\!]$ to B	
2		Compute $[\![Z]\!] = [\![S^K]\!]X\Theta$ and $[\![S^K X]\!] = [\![S^K]\!]X$, send $[\![Z]\!] + [\![R_{B1}]\!]$ and $[\![S^K X]\!] + [\![R_{B2}]\!]$ to A
3	Decrypt $[\![Z]\!] + [\![R_{B1}]\!]$ and $[\![S^K X]\!] + [\![R_{B2}]\!]$, send $Z + R_{B1}$ and $S^K X + R_{B2}$ to B	
4		Compute $\frac{\partial \mathcal{L}}{\partial \Theta} = \frac{1}{n}(\sigma(Z) - Y)^T (S^K X)$ and update Θ

Table 2: Model training procedure of FedSGC-VAF

assume that the two parties hold different but partially overlapping users, which can be identified using their IDs. Similar to other vertical federated learning algorithms like [Cheng *et al.*, 2019], we align the data across the two parties using a privacy-preserving protocol of inter-database intersections [Liang and Chawathe, 2004]. Then, we collaboratively train the FedSGC model without violating privacy. The main steps of FedSGC-VAT under the discussed settings is illustrated in Table 1. The major idea is to encrypt the sensitive data using AHE. In step 3 of Table 1, party B can multiply X with the encrypted information from A as HE allows scalar multiplication and cipher-text addition. Step 1 to 5 will be performed iteratively until some convergence conditions are satisfied. If A needs to make inference on record i, it can notify party B and execute step 1 and 2 with the corresponding X_i and S_i^K.

Case 2: FedSGC with Vertical Active Feature party (FedSGC-VAF) The second setting is similar to case 1 except the first party A owns the normalized adjacency matrix S only; and the second party B owns the feature X and target label Y (Figure 1b). Similar with case 1, we assume that there are certain overlapping data between A and B and align the data across using a privacy-preserving protocol of inter-database intersections [Liang and Chawathe, 2004]. Then, the two parties collaboratively train the FedSGC model without leaking the S, X and Y to the other party. The major steps is illustrated in Table 2. At inference time, B can send the encrypted Z_i using the corresponding X_i to A for decryption in Step 2-3 (Table 2) and compute the prediction $\sigma(Z_i)$ accordingly.

Case 3: FedSGC with Horizontal Node party (FedSGC-HN) In this setting, every node in the graph owns its own feature X_i, target label Y_i and connections to its 1-hop neighbour $N(i)$, where $N(i)$ denotes the set of node IDs of the 1-hop neighbours from node i (Figure 1c). Unlike the conventional horizontal federated learning, the nodes cannot compute individual gradients and aggregate the parameters as there is a feature propagation step, which relies on the information from other parties. To achieve this, we propose to perform message passing through encrypted communications between nodes. We also employ a parameter server for parameter aggregation and message decryption. The major steps is illustrated in Table 3. In Step 2-3, each node is performing message passing by sending the encrypted transformed feature and encrypted feature to neighbour nodes. Since each node does not know the degree information of its neighbouring node, it aggregates the messages by dividing the summed encrypted node features with its node degree $|N(i)| + 1$, which is slightly different from the original SGC using $\sqrt{|N(i)| + 1} \cdot \sqrt{|N(j)| + 1}$ as the denominator for normalization between node i and j. By repeating Step 2 and 3 for K rounds, the information is passed among K-hop neighbours, which is same as computing $S^K X\Theta$ in central-

Step	Node i (with $X_i, Y_i, N(i)$)	Parameter Server P
0		Create encryption key pairs, initialize K and Θ, distribute the public key, K and Θ to all nodes
1	Compute $[\![H_i\Theta]\!]$ and $[\![H_i]\!]$	
2	Send $[\![H_i\Theta]\!]$ and $[\![H_i]\!]$ to node $j \in N(i)$	
3	Receive $\{[\![H_j\Theta]\!]\}$ and $\{[\![H_j]\!]\}$ for $j \in N(i)$ Update $[\![H_i\Theta]\!] = \frac{1}{\lvert N(i)+1\rvert}\sum_{j\in N(i)\cap\{i\}}[\![H_j\Theta]\!]$ Update $[\![H_i]\!] = \frac{1}{\lvert N(i)+1\rvert}\sum_{j\in N(i)\cap\{i\}}[\![H_j]\!]$ Repeat Step 2 and 3 for K rounds	
4	Send $[\![H_i\Theta]\!] + [\![R_i]\!]$ to S	
5		Decrypt $[\![H_i\Theta]\!] + [\![R_i]\!]$ and send $H_i\Theta + R_i$ to Node i
6	Compute $[\![\frac{\partial\mathcal{L}}{\partial\Theta}]\!]_i = (\sigma(H_i\Theta) - Y_i)^T[\![H_i]\!]$ Send $[\![\frac{\partial\mathcal{L}}{\partial\Theta}]\!]_i$ to P	
7		Decrypt $[\![\frac{\partial\mathcal{L}}{\partial\Theta}]\!]_i$ and compute $\frac{\partial\mathcal{L}}{\partial\Theta} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\partial\mathcal{L}}{\partial\Theta}\right)_i$ Update Θ and distribute Θ to all nodes

Table 3: Model training procedure of FedSGC-HN

ized manner. For the inference prediction, a node can perform Step 2 to 5 and compute $\hat{Y}_i = \sigma(H_i\Theta)$ as the prediction.

4 Experiments and Results

Datasets In this section, we empirically demonstrate that FedSGC achieves similar accuraccies with SGC trained in centralized settings, whilst maintaining the the security and privacy under federated learning condition. We follow the experiment setup in [Wu *et al.*, 2019] and conduct the experiments on CiteSeer, Cora and PubMed citation networks datasets transductively [Sen *et al.*, 2008]. The datatset statistics are summarized in Table 4.

Dataset	Nodes	Edges	Classes	Train/Test
Cora	2,708	5,429	7	140/1,000
Citeseer	3,327	4,732	6	120/1,000
Cora	19,717	444,338	3	60/1,000

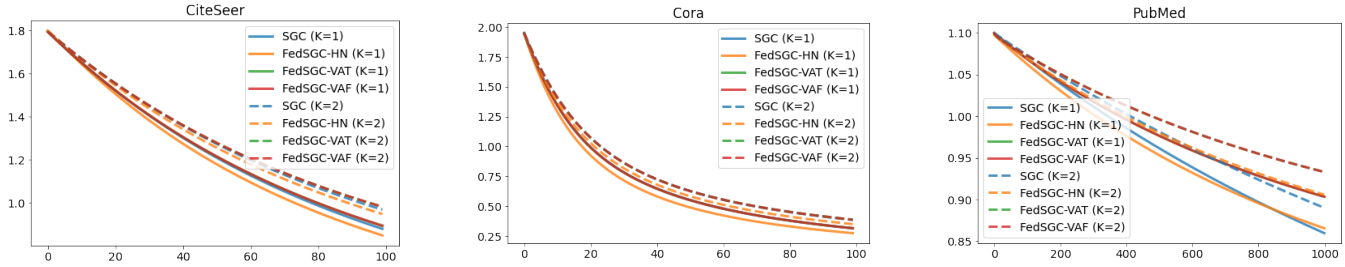
Table 4: Dataset statistics of the the citation network datasets

Experiment Settings For the SGC baseline and FedSGC, we train the models using SGD. We tune the learning rate, weight decay and the number of epochs on each dataset using the hold-out validation set available publicly. We compare the training and test accuracy with the SGC baseline for $K = 1$ and $K = 2$.

Results First, we compare the convergence between non-federated SGC and FedSGC under different federated settings (case 1-3). In Figure 2, the training loss profile of SGC is very close to that in FedSGC. The convergence behaviour of SGC and FedSGC are very much alike. Then, we also investigate the prediction performance between SGC and FedSGC. We compare their final training and test accuracy in Table 5. From the results, we verify that the prediction performance

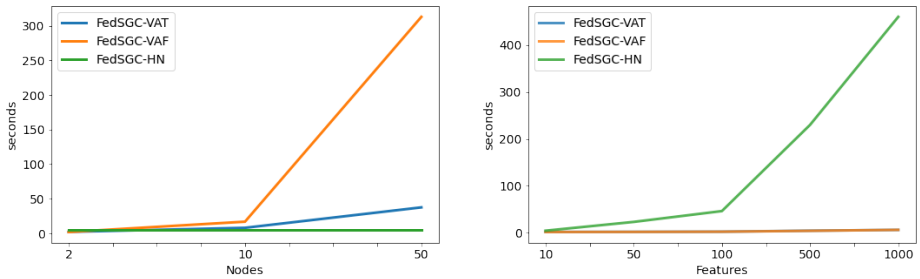
Dataset	Model	Training Accuracy	Test Accuracy
Cora ($K = 1$)	SGC	0.99	0.78
	FedSGC-VAT	0.99	0.79
	FedSGC-VAF	0.99	0.79
	FedSGC-HN	0.99	0.79
Cora ($K = 2$)	SGC	0.99	0.82
	FedSGC-VAT	0.98	0.82
	FedSGC-VAF	0.99	0.82
	FedSGC-HN	0.99	0.82
CiteSeer ($K = 1$)	SGC	0.95	0.70
	FedSGC-VAT	0.96	0.70
	FedSGC-VAF	0.96	0.70
	FedSGC-HN	0.95	0.71
CiteSeer ($K = 2$)	SGC	0.92	0.72
	FedSGC-VAT	0.92	0.72
	FedSGC-VAF	0.92	0.72
	FedSGC-HN	0.90	0.72
PubMed ($K = 1$)	SGC	0.93	0.75
	FedSGC-VAT	0.93	0.75
	FedSGC-VAF	0.93	0.75
	FedSGC-HN	0.93	0.74
PubMed ($K = 2$)	SGC	0.89	0.74
	FedSGC-VAT	0.88	0.73
	FedSGC-VAF	0.88	0.73
	FedSGC-HN	0.87	0.72

Table 5: Training and test accuracy for SGC, FedSGC-VAT, FedSGC-VAF and FedSGC-HN on citation networks datasets for $K = 1, 2$



(a) Citeseer (b) Cora (c) PubMed

Figure 2: Illustration of the training loss profile when training SGC and FedSGC on citation networks for $K = 1, 2$



(a) (b)

Figure 3: Illustration of the scalability of FedSGC for different numbers of (a) nodes and (b) features

of FedSGC is closely aligned with the SGC model that is being trained in centralized manner. Under the vertical settings, FedSGC-VAT and FedSGC-VAF produces the accuracy within 1% with the SGC baseline. For horizontal setting, the accuracy differences are within 2%. One possible reason of the slight mis-match may be the different normalization used in the horizontal setting.

Scalability Our framework results in a learned model closely aligned with the centralized model by design. The model updates performed are mostly the same with the only differences being the message passing protocol. Privacy is also guaranteed by ensuring raw features or graph values are not passed in plain text between the parties. The main limitation however is the bottleneck from performing encryption. FedSGC applies encryption over the raw, transformed feature or the adjacency matrix depending on the setting. We test the scalability of FedSGC by comparing the execution time per model update for different numbers of training nodes n and feature sizes f (see Figure 3). Our results show that FedSGC-HN scales well with the number of training nodes as the encryption can be done in parallel within each node parties. However, as the encryption is applied to the feature vector and the transformed feature. There are $f+c$ parameters to be encrypted with c denoting the number of classes. The execution time is longer if the feature size is large. FedSGC-VAT scales well with increased number of feature size, with nc values to be encrypted. For FedSGC-VAF, there are n^2 values to be encrypted in the adjacency matrix, thus requiring a longer time when the number of training nodes is large. As the adjacency matrix is usually sparse, it is beneficial to investigate if there is a more efficient way to encrypt the sparse matrix to speed up the process.

5 Analysis

We analyse the security of FedSGC under the assumption that all federated parties are semi-honest. For FedSGC-VAT, the passive feature party only pass the transformed feature $X\Theta$ to the active topology party. From the active topology party, all the transmitted information are encrypted. Both parties cannot reconstruct the data from others. In case 2, all the communications in FedSGC-VAF are encrypted. Even party B knows the value of $Z = S^K X\Theta$, it cannot reconstruct back S^K as $X\Theta$ is typically non-invertable. In FedSGC-HN, the communications between horizontal node parties are encrypted. Upon decryption, each node will only learn the aggregated transformed feature (i.e. $H_i\Theta$ in Step 6 of FedSGC-HN). During training, the server only knows the parameter Θ and the gradient information from the node parties.

6 Conclusion

In this study, we propose FedSGC-VAT, FedSGC-VAT and FedSGC-HN to solve three federated learning settings in graph data. We demonstrate through experiments that FedSGC leads to performance closely aligned to the non-federated model trained in centralized manner. We also show that the privacy and security can be properly preserved. However, FedSGC requires a longer time to perform encryption. Also, the proposed FedSGC workflows may not transfer well to other more complicated GNN models. Nevertheless, the feasibility of such a framework has been demonstrated, where a SGC problem can be easily adapted to a federated approach.

References

[Acar *et al.*, 2018] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.*, 51(4):79:1–79:35, 2018.

[Cheng *et al.*, 2019] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, and Qiang Yang. Secureboost: A lossless federated learning framework. *arXiv preprint arXiv:1901.08755*, 2019.

[Fout, 2017] Alex M Fout. *Protein interface prediction using graph convolutional networks*. PhD thesis, Colorado State University, 2017.

[Gascón *et al.*, 2016] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Secure linear regression on vertically partitioned datasets. *IACR Cryptol. ePrint Arch.*, 2016:892, 2016.

[Hardy *et al.*, 2017] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *CoRR*, abs/1711.10677, 2017.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[Konečnỳ *et al.*, 2016a] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

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

[Li *et al.*, 2017] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[Liang and Chawathe, 2004] Gang Liang and Sudarshan S. Chawathe. Privacy-preserving inter-database operations. In *Intelligence and Security Informatics, Second Symposium on Intelligence and Security Informatics, ISI 2004*, volume 3073, pages 66–82. Springer, 2004.

[McMahan *et al.*, 2016] H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629*, 2016.

[McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54, pages 1273–1282. PMLR, 2017.

[Paillier, 1999] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 223–238. Springer, 1999.

[Rivest *et al.*, 1978] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[Sen *et al.*, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93, 2008.

[Wang *et al.*, 2017] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 889–898, 2017.

[Wu *et al.*, 2019] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR, 2019.

[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.

[Yang *et al.*, 2019] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2), 2019.

[Ying *et al.*, 2018] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

[Zhang and Chen, 2018] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *arXiv preprint arXiv:1802.09691*, 2018.

[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.

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