# FedSGC: Federated Simple Graph Convolution for Node Classification

Tsz-Him Cheung , Weihang Dai and Shuhan Li

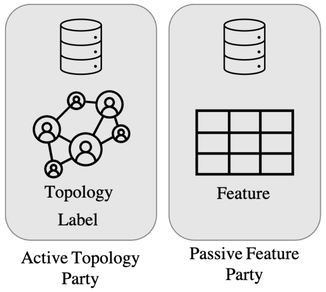
The Hong Kong University of Science and Technology

## Introduction

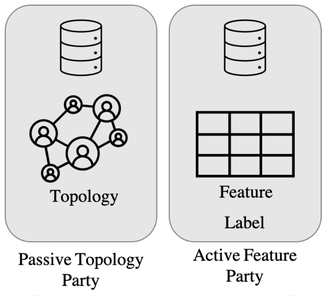
This paper proposes three federated settings in learning graph neural network: (1) vertical FL with one active topology party and one passive feature party; (2) vertical FL with one passive topology party and one active feature party; (3) horizontal FL where each graph node is a party. Different from the conventional vertical and horizontal FL, we consider the graph topology as another form of "feature" that can be sensitive and owned by different parties, which is more suitable for real-life applications, for example in e-commerce.

To tackle the three graph FL settings, we introduce FedSGC, which incorporates federated learning techniques to train the Simple Graph Convolution (SGC) model. We demonstrate that the prediction performance of FedSGC is closely aligned with the non-federated model trained in centralised manner.

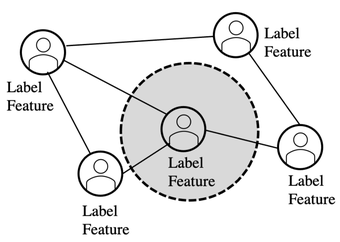
## Problem Setting




**Case 1:** Two parties collaborate to train a SGC model in the 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 the 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 the horizontal federated learning setting. Each node party owns its feature, the connections to its 1-hop neighbour and its label.

## Methodology

### Homomorphic Encryption

Homomorphic Encryption [1] allows computation over encrypted ciphertext while keeping the computed results as encrypted and same with the results computed in plaintext. FedSGC uses additively homomorphic encryption (AHE), which allows certain additive operations to be carried out between ciphertexts while maintaining certain degrees of computation efficiency. Specifically, the AHE we used requires:

$$[[u]] + [[v]] = [[u + v]]$$
$$a \cdot [[u]] = [[a \cdot u]]$$

### Simple Graph Convolution (SGC)

The simple graph convolution [2] linearises the computation of graph convolution networks (GCNs) by removing the nonlinear activations in GCN and computes the prediction with: $\hat{Y} = \sigma(S^K X \Theta)$, where $S, X, \Theta$ denote the normalised adjacency matrix with self-loop, the feature and the linear transformation weight matrix, respectively. In classification task, the gradient of the learnable parameters can be expressed as the linear product between the topology, the data feature and the prediction labels, which allow the use of AHE to protect the data exchange between multiple parties while training the model.

### FedSGC

**Case 1: FedSGC with Vertical Active Topology party (FedSGC-VAT)**

| 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 $\Theta$ |
| 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 $\Theta$ |

**Case 2: FedSGC with Vertical Active Feature party (FedSGC-VAF)**

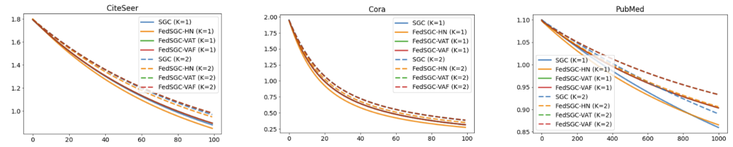
| Step | Passive Topology Party A (with $S$) | Active Feature Party B (with $X, Y$) |
|---|---|---|
| 0 | Create encryption key pairs and send the public key to B, initialize $K$ | Initialize $\Theta$ |
| 1 | Compute $S^K$ and send $[\![S^K]\!]$ to B | |
| 2 | | Compute $[\![Z]\!] = [\![S^K]\!] X\Theta$ 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 $\Theta$ |

**Case 3: FedSGC with Horizontal Node party (FedSGC- HN)**

| Step | Node $i$ (with $X_i, Y_i, N(i)$) | Parameter Server P |
|---|---|---|
| 0 | | Create encryption key pairs, initialize $K$ and $\Theta$, distribute the public key, $K$ and $\Theta$ 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 $j \in N(i)$<br>Update $[\![H_i\Theta]\!] = \frac{1}{\lvert N(i)+1 \rvert} \sum_{j \in N(i) \cap \{i\}} [\![H_j\Theta]\!]$<br>Update $[\![H_i]\!] = \frac{1}{\lvert N(i)+1 \rvert} \sum_{j \in N(i) \cap \{i\}} [\![H_j]\!]$<br>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 $[\![\frac{\partial \mathcal{L}}{\partial \Theta}]\!]_i = (\sigma(H_i\Theta) - Y_i)^T [\![H_i]\!]$<br>Send $[\![\frac{\partial \mathcal{L}}{\partial \Theta}]\!]_i$ to P | |
| 7 | | Decrypt $[\![\frac{\partial \mathcal{L}}{\partial \Theta}]\!]_i$ and compute $\frac{\partial \mathcal{L}}{\partial \Theta} = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\partial \mathcal{L}}{\partial \Theta}\right)_i$<br>Update $\Theta$ and distribute $\Theta$ to all nodes |

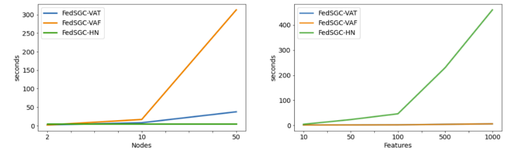
## Experiment

Our experiments show that the proposed FedSGC performs similarly with the SGC model trained in centralised manner in terms of convergence and classification performance. The major limitation is the bottleneck from performing encryption. Specifically, FedSGC applies AHE over the raw, transformed feature or the adjacency matrix depending on the settings. It may affect the scalability when the feature size or the number of nodes are large.




(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)

## 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 centralised manner. We also show that the privacy and security can be properly preserved.

### References

[1] 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.

[2] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Wein- berger. 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.