A FedGCN Training Algorithm

Algorithm 1 `FedGCN` Federated Training for Graph Convolutional Network

`// Pre-Training Communication Round`
for each client $k \in [K]$ **do in parallel**
  Send $[\![\{\sum_{j\in\mathcal{N}_i} \mathbb{I}_k(c(j)) \cdot \boldsymbol{A}_{ij}\boldsymbol{x}_j\}_{i\in\mathcal{V}_k}]\!]$ to the server
end
`// Server Operation`
for $i \in \mathcal{V}$ **do in parallel**
  $[\![\sum_{j\in\mathcal{N}_i} \boldsymbol{A}_{ij}\boldsymbol{x}_j]\!] = \sum_{d=1}^{C} [\![\sum_{j\in\mathcal{N}_i} \mathbb{I}_k(c(j)) \cdot \boldsymbol{A}_{ij}\boldsymbol{x}_j]\!]$
end
for each client $k \in [K]$ **do in parallel**
  if 1-hop **then**
    Receive $[\![\{\sum_{j\in\mathcal{N}_i} \boldsymbol{A}_{ij}\boldsymbol{x}_j\}_{i\in\mathcal{V}_k}]\!]$ and decrypt it
  end
  if 2-hop **then**
    Receive $[\![\{\sum_{j\in\mathcal{N}_i} \boldsymbol{A}_{ij}\boldsymbol{x}_j\}_{i\in\mathcal{N}_{\mathcal{V}_k}}]\!]$ and decrypt it
  end
end
`// Training Rounds`
for $t = 1, \ldots, T$ **do**
  for each client $k \in [K]$ **do in parallel**
    Receive $[\![\boldsymbol{w}^{(t)}]\!]$ and decrypt it
    Set $\boldsymbol{w}_k^{(t,0)} = \boldsymbol{w}^{(t)}$,
    for $e = 1, \ldots, E$ **do**
      Set $\boldsymbol{g}_{\boldsymbol{w}_k}^{(t,e)} = \nabla_{\boldsymbol{w}_k} f_k(\boldsymbol{w}_k^{(t,e-1)}; \mathcal{G}_k)$
      $\boldsymbol{w}_k^{(t,e)} = \boldsymbol{w}_k^{(t,e-1)} - \eta\, \boldsymbol{g}_{\boldsymbol{w}_k}^{(t,e)}$ `// Update Parameters`
    end
    Send $[\boldsymbol{w}_k^{(t,E)}]\!]$ to the server
  end
  `// Server Operations`
  $[\![\boldsymbol{w}^{(t+1)}]\!] = \frac{1}{K}\sum_{d=1}^{C} [\![\boldsymbol{w}_k^{(t,E)}]\!]$ `// Update Global Models`
  Broadcast $[\![\boldsymbol{w}^{(t+1)}]\!]$ to local clients
end

B Applications of FedGCN

In this section, we discuss three important yet challenging applications of federated graph learning. Compared with prior works, our FedGCN can overcome the challenges of each application setting without sacrificing accuracy.

B.1 Millions of Clients with Small Graphs

With the development of IoT (Internet-of-Things) devices, many people own several devices that can collect, process, and communicate data (e.g., Mobile phones, smartwatches, or computers). Recently, smart home devices (e.g. cameras, light bulbs, and smart speakers) have also been adopted by millions of users, e.g., to monitor home security, the health care of senior citizens and infants, and package delivery. These mobile devices, and the applications that run on them, can also have interactions and connect with these IoT devices, e.g., unlocking the front door triggers the living room camera.

All these devices and applications are connected and form a graph, in which the devices/applications are nodes and their interactions are edges. The information of local devices or applications (node features) can then be very privacy sensitive, e.g., it may include video recording, accurate user location, and other information about users' personal habits. Federated training keeps the data localized to maintain privacy, while cross-client edges affect the federated training performance with privacy leakage. Here we take a “client” to mean a single user, who may own several devices or applications (i.e., nodes in the graph). Millions of devices across multiple users can then be connected, although each user (client) owns only a few devices, which means there are many edges across clients. The huge amount of cross-client edges can then seriously affect federated training's performance. The condition becomes more serious with the co-optimization of heterogeneous data distribution with limited local data.

Our FedGCN can significantly improve both the convergence time and model accuracy, since it does not have information loss regardless of the number of clients. Since the number of cross-client edges increases with the number of clients, prior methods that ignore these edges or communicate (some of) their information in every round respectively face more information loss and communication costs.

B.2 A Few Clients with Large Graphs

Due to privacy regulations (e.g., GDPR in Europe) across countries, some data, e.g., that collected by Internet services like social networks, needs to be localized in each country or region. Here, each country represents a client and “nodes” might be citizens of that country who use a particular service, g., users of a social network. Users in different countries (at different clients) may then interact, creating a cross-client edge. In this case, a single large country might be able to train models with sufficient performance using only it's users' information, but some small countries might find it hard to train a good model as they have fewer constituent nodes. Federated learning for training a global model across countries, i.e., cross-silo federated learning, can help to train models in this setting, and there are relatively few cross-client edges compared to in-client edges. For example, in social networks, edges represent the connections between users, and users are more closely connected within a country. However, the small amount of cross-country (cross-client) edges might seriously affect the model performance since these edges can be more important for decision-making than the in-client edges. For example, for anomaly detection on payment records, cross-country transactions are the key to detecting international money laundering and fraud, and ignoring these edges makes it impossible to detect these behaviors. FedGCN can take these edges into account and the trainers can decide if they want these edges based on the edge utility for their tasks.

FedGCN only communicates accumulated and homomorphically encrypted neighbor information at the initial round with better privacy guarantees and can also add differential privacy to better fit privacy regulations.

B.3 Multi-step Distributed Training

In distributed training, the main focus is to train a model with fast computation time and high accuracy, utilizing the resources of multiple computing servers. Privacy is not a concern in this case. FedGCN requires much less communication cost compared with distributed training methods (e.g., BDS-GCN Wan et al. (2022)) since FedGCN only requires pre-training communication. Moreover, FedGCN suggests that non-i.i.d. data distributions can further reduce the communication, a since non-i.i.d. partition results in fewer cross-client edges. FedGCN can first partition the graph to non-i.i.d. and perform precomputation to minimize the communication cost while maintaining model accuracy.

C Future Directions

Although the FedGCN can overcome the challenges mentioned above, it mainly works on training accumulation-based models like GCN and GraphSage. There are several open problems in federated graph learning that need to be explored.

C.1 Federated Training of Attention-based GNNs

Attention-based GNNs like GAT (Graph attention network) require calculating the attention weights of edges during neighbor feature aggregation, where the attention weights are based on the node features on both sides of edges and attention parameters. The attention parameters are updated at every training iteration and cannot be simply fixed at the initial round. How to train attention-based

GNNs in a federated way with high performance and privacy guarantees is an open challenge and promising direction.

C.2 Neighbor Node and Feature Selection to Optimize System Performance

General federated graph learning optimizes the system by only sharing local models, without utilizing cross-device graph edge information, which leads to less accurate global models. On the other hand, communicating massive additional graph data among devices introduces communication overhead and potential privacy leakage. To save the communication cost without affecting the model performance, one can select key neighbors and neighbor features to reduce communication costs and remove redundant information. For privacy guarantee, if there is one neighbor node, it can be simply dropped to avoid private data communication. FedGCN can be extended by using selective communication in its pre-training communication round.

C.3 Integration with K-hop Linear GNN Approximation methods

To speed up the local computation speed, L-hop Linear GNN Approximation methods use precomputation to reduce the training computations by running a simplified GCN ($\boldsymbol{A}^L\boldsymbol{X}\boldsymbol{W}$ in SGC Wu et al. (2019), $[\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}, \boldsymbol{A}^2\boldsymbol{X}\boldsymbol{W}, \ldots, \boldsymbol{A}^L\boldsymbol{X}\boldsymbol{W}]$ in SIGN Frasca et al. (2020), and $\boldsymbol{\Pi}\boldsymbol{X}\boldsymbol{W}$ in PPRGo Bojchevski et al. (2020) where $\boldsymbol{\Pi}$ is the pre-computed personalized pagerank), but the communication cost is not reduced if we perform these methods alone. They are thus a complementary approach for efficient GNN training. FedGCN (2-hop, 1-hop) changes the model input ($\boldsymbol{A}$ and $\boldsymbol{X}$) to reduce communication in the FL setting, but the GCN model itself is not simplified. FedGCN can incorporate these methods to speed up the local computation, especially in constrained edge devices.

D Background and Preliminaries

D.1 Federated Learning

Federated learning was first proposed by McMahan et al. (2017), who build decentralized machine learning models while keeping personal data on clients. Instead of uploading data to the server for centralized training, clients process their local data and occasionally share model updates with the server. Weights from a large population of clients are aggregated by the server and combined to create an improved global model.

The FedAvg algorithm McMahan et al. (2017) is used on the server to combine client updates and produce a new global model. At training round t, a global model $\boldsymbol{w}^{(t)}$ is sent to K client devices.

At each local iteration e, every client k computes the gradient, $\boldsymbol{g}_{\boldsymbol{w}_k}^{(t,e)}$, on its local data by using the current model $\boldsymbol{w}_k^{(t,e-1)}$. For a client learning rate η, the local client update at the e-th local iteration, $\boldsymbol{w}_k^{(t,e)}$, is given by

$$\boldsymbol{w}_k^{(t,e)} \leftarrow \boldsymbol{w}_k^{(t,e-1)} - \eta \boldsymbol{g}_{\boldsymbol{w}_k}^{(t,e)}. \tag{1}$$

After E local iterations, the server then does an aggregation of clients' local models to obtain a new global model,

$$\boldsymbol{w}^{(t+1)} = \frac{1}{K}\sum_{d=1}^{C} \boldsymbol{w}_k^{(t,E)}. \tag{2}$$

The process then advances to the next training round, $t+1$.

D.2 Graph Convolutional Network

A multi-layer Graph Convolutional Network (GCN) (Kipf and Welling, 2016) has the layer-wise propagation rule

$$\boldsymbol{H}^{(l+1)} = \phi(\boldsymbol{A}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}). \tag{3}$$

The weight adjacency matrix $\boldsymbol{A}$ can be normalized or non-normalized given the original graph, and $\boldsymbol{W}^{(l)}$ is a layer-specific trainable weight matrix. The activation function is ϕ, typically ReLU (rectified linear units), with a softmax in the last layer for node classification. The node embedding matrix in the l-th layer is $\boldsymbol{H}^{(l)} \in \mathbb{R}^{N\times d}$, which contains high-level representations of the graph nodes transformed from the initial features; $\boldsymbol{H}^{(0)} = \boldsymbol{X}$.

In general, for a GCN with L layers of the form 3, the output for node i will depend on neighbors up to L steps away. We denote this set by $\mathcal{N}_i^L$ as L-hop neighbors of i. Based on this idea, the clients can first communicate the information of nodes. After the communication of information, we can then train the model.

D.3 Stochastic Block Model

For positive integers C and N, a probability vector $\boldsymbol{p} \in [0,1]^C$, and a symmetric connectivity matrix $\boldsymbol{B} \in [0,1]^{C\times C}$, the SBM defines a random graph with N nodes split into C classes. The goal of a prediction method for the SBM is to correctly divide nodes into their corresponding classes, based on the graph structure. Each node is independently and randomly assigned a class in $\{1, ..., C\}$ according to the distribution $\boldsymbol{p}$; we can then say that a node is a "member" of this class. Undirected edges are independently created between any pair of nodes in classes c and d with probability $\boldsymbol{B}_{cd}$, where the (c, d) entry of $\boldsymbol{B}$ is

$$\boldsymbol{B}_{cd} = \begin{cases} \alpha, & c = d \\ \mu\alpha, & c \neq d, \end{cases} \tag{4}$$

for $\alpha \in (0,1)$ and $\mu \in (0,1)$, implying that the probability of an edge forming between nodes in the same class is α (which is the same for each class) and the edge formation probability between nodes in different classes is $\mu\alpha$.

Let $\boldsymbol{Y} \in \{0,1\}^{N\times C}$ denotes the matrix representing the nodes' class memberships, where $\boldsymbol{Y}_{ic} = 1$ indicates that node i belongs to the c-th class, and is 0 otherwise. We use $\boldsymbol{A} \in \{0,1\}^{N\times N}$ to denote the (symmetric) adjacency matrix of the graph, where $\boldsymbol{A}_{ij}$ indicates whether there is a connection (edge) between node i and node j. From our node connectivity model, we find that given $\boldsymbol{Y}$, for $i < j$, we have

$$\boldsymbol{A}_{ij} | \{\boldsymbol{Y}_{ic} = 1, \boldsymbol{Y}_{jd} = 1\} \backsim \mathrm{Ber}(\boldsymbol{B}_{cd}), \tag{5}$$

where $\mathrm{Ber}(p)$ indicates a Bernoulli random variable with parameter p. Since all edges are undirected, $\boldsymbol{A}_{ij} = \boldsymbol{A}_{ji}$. We further define the connection probability matrix $\boldsymbol{P} = \boldsymbol{Y}\boldsymbol{B}\boldsymbol{Y}^T \in [0,1]^{N\times N}$, where $\boldsymbol{P}_{ij}$ is the connection probability of node i and node j and $\mathbb{E}[\boldsymbol{A}] = \boldsymbol{P}$.

E Training Configuration

E.1 Statistics of Datasets

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Ogbn-Arxiv	169,343	1,166,243	128	40
Ogbn-Products	2,449,029	61,859,140	100	47

Table 1: Statistics of datasets.

E.2 Experiment Hyperparameters

For Cora and Citeseer, we use a two-layer GCN with ReLU activation for the first and Softmax for the second layer, as in Kipf and Welling (2016). There are 16 hidden units. A dropout layer between the two GCN layers has a dropout rate of 0.5. We use 300 training rounds with the SGD optimizer for all settings with a learning rate of 0.5, L2 regularization 5×10^4, and 3 local steps per round for federated settings. For the OGBN-Arxiv dataset, we instead use a 3-layer GCN with 256 hidden units and 600 training rounds. For the OGBN-Products dataset, we use 2-layer GraphSage, 256 hidden units, and 450 training rounds. All settings are the same as the papers Kipf and Welling (2016); Hu et al. (2020). The local adjacency matrix is normalized by $\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$ when using GCN. We evaluate the local test accuracy given the local graph $\mathcal{G}_k$ and get the average test accuracy of all clients as the global test accuracy. We set the number of clients to 10 and averaged over 10 experiment runs.

E.3 Computation Resource

Experiments are done in a p3d.16xlarge instance with 8 GPUs (32GB memory for each GPU) and 10 g4dn.xlarge instances (16GB GPU memory in each instance). One run of the OGBN-Products experiment can take 20 minutes due to full-batch graph training.

F Communication Cost and Tradeoffs

Data Distribution	0-hop	1-hop	2-hop
Generic Graph	0	$\sum_{i\in\mathcal{V}}\lvert c(\mathcal{N}_i)\rvert d + Nd$	$\sum_{i\in\mathcal{V}}\lvert c(\mathcal{N}_i)\rvert d + \sum_{k=1}^{K}\lvert\mathcal{N}_{\mathcal{V}_k}\rvert d$
Non-i.i.d. (SBM)	0	$(c_\mu+2)Nd$	$2(c_\mu+1)Nd$
Partial-i.i.d. (SBM)	0	$(c_\alpha p+c_\mu+2)Nd$	$2(c_\alpha p+c_\mu+1)Nd$
i.i.d. (SBM)	0	$(c_\alpha+c_\mu+2)Nd$	$2(c_\alpha+c_\mu+1)Nd$

Table 2: Communication costs. $|.|$ denotes the size of the set and $\sum_{i\in\mathcal{V}}|c(\mathcal{N}_i)|d$ is the cost of the message that the server received from all clients, where $c(\mathcal{N}_i)$ denotes the set of clients storing the neighbors of node i. Communication cost increases with the i.i.d control parameter $\boldsymbol{p}$. 2-hop communication has around twice the cost of 1-hop communication.

In this appendix, we examine the communication cost, and the resulting convergence-communication tradeoff, of FedGCN. As for the convergence analysis in Section **??**, we derive communication costs for general graphs and then more interpretable results for the SBM model.

Proposition F.1. *(Communication Cost for FedGCN) For L-hop communication of GCNs with number of layers $\geq L$, the size of messages from K clients to the server in a generic graph is*

$$\sum_{i\in\mathcal{V}}|c(\mathcal{N}_i)|d + \sum_{k=1}^{K}|\mathcal{N}_k^{L-1}|d, \tag{6}$$

where $c(\mathcal{N}_i)$ denotes the set of clients storing the neighbors of node i.

For a better understanding of the above form, Table 2 gives the approximated (assuming $\alpha, \mu << 1$) size of messages between clients for i.i.d. and non-i.i.d. data, for generic graphs and an SBM with N nodes and d-dimensional node features. Half the partial i.i.d. nodes are chosen in the i.i.d. and half the non-i.i.d. settings.

Appendix I proves this result. In the non-i.i.d. setting, most nodes with the same labels are stored in the same client, which means there are much fewer edges linked to nodes in the other clients than in the i.i.d. setting, incurring much less communication cost (specifically, $c_\alpha Nd$ fewer communications) for 1- and 2-hop FedGCN. Note that communication costs vary with N but not K, the number of clients, as clients communicate directly with the server and not with each other.

Combining Table **??**'s and Table 2's results, we observe i.i.d. data reduces the gradient variance but increases the communication cost, while the non-i.i.d. setting does the opposite. Approximation methods via one-hop communication then might be able to balance the convergence rate and communication. We experimentally validate this intuition in Section **??**'s results, as well as the next appendix section.

G Additional Experimental Results

G.1 Validation of Theoretical Analysis on Cora Dataset

We validate the qualitative results in main theory and F.1 on the Cora dataset. As shown in Figure 1, 0-hop FedGCN does not need to communicate but requires high convergence time. One- and 2-hop FedGCN have similar convergence time, but 1-hop FedGCN needs much less communication. The right graph in Figure 1 shows Table **??**'s gradient norm bound for the Cora dataset. We expect these to qualitatively follow the same trends as we increase the fraction of i.i.d. data, since from Theorem **??** the convergence time increases with $\|\nabla f_k(\boldsymbol{w}_k) - \nabla f(\boldsymbol{w})\|$. FedGCN (2-hop) and FedGCN (0-hop), as we would intuitively expect, respectively decrease and increase: as the data becomes more i.i.d., FedGCN (0-hop) has more information loss, while FedGCN (2-hop) gains more useful information

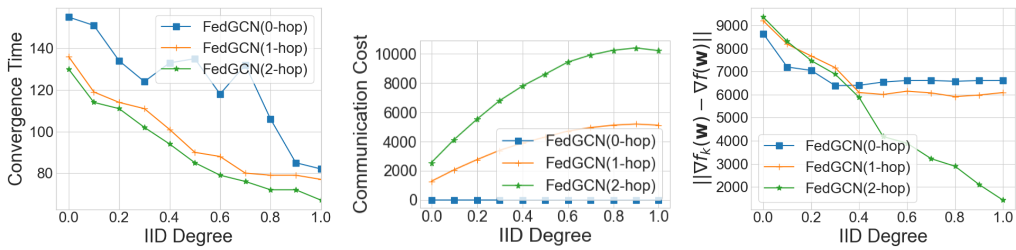

Figure 1: Convergence time (left), communication cost (middle) on Cora, and theoretical convergence upper bound (right, Table **??**). FedGCN (1-hop) balances convergence and communication.

from cross-client edges. Federated learning also converges faster for i.i.d. data, and we observe that FedGCN (0-hop)'s increase in convergence time levels off for $> 80\%$ i.i.d. data.

G.2 Homomorphic Encryption Microbenchmarking

Scheme	**Cheon-Kim-Kim-Song (CKKS)**
ring dimension	4096
security level	HEStd_128_classic
multi depth	1
scale factor bits	30

Table 3: HE Scheme Parameter Configuration On PALISADE. Multi depth is configured to be 1 for optimal (minimum) maximum possible multiplicative depth in our evaluation.

We implement our HE module using the HE library PALISADE (v1.10.5) PALISADE (2020) with the cryptocontext paramteters configuration as in Table 3. In our paper, we evaluate the real-number HE scheme, i.e., the Cheon-Kim-Kim-Song (CKKS) scheme Cheon et al. (2017).

Array Size	Plaintext (Bool)	Plaintext (Long & Double)	CKKS	CKKS (Boolean Packing)
1k	1 kB	8 kB	266 kB	266 kB
10k	10 kB	80 kB	798 kB	266 kB
100k	100 kB	800 kB	7 MB	1 MB
1M	1 MB	8 MB	70 MB	8 MB
100M	100 MB	800 MB	7 GB	793 MB
1B	1 GB	8 GB	70 GB	8 GB

Table 4: Communication Cost Comparison between Plaintext and Encryption: Plaintext files are numpy arrays with pickle and ciphertext files are generated under CKKS.

In our framework, neighboring features (long integers, int64) are securely aggregated under the BGV scheme and local model parameters (double-precision floating-point, float64) are securely aggregated under the CKKS scheme. The microbenchmark results of additional communication overhead can be found in Table 4. In general, secure computation using HE yields a nearly 15-fold increase of communicational cost compared to insecure communication in a complete view of plaintexts. However, with our Boolean Packing technique, the communication overhead only **doubles** for a large-size array.

H Convergence Proof

We first give an example of a 1-layer GCN, then we mainly analyze a 2-layer GCN, which is the most common architecture for graph neural networks. The intuition of the theory is bounding the difference between the local gradient and global gradient in non-i.i.d settings. Our analysis also fits any layers of GCN and GraphSage.

H.1 Convergence Analysis of 1-layer GCNs

We first get the gradient of 1-layer GCNs in centralized, 0-hop and 1-hop cases. Then we provide bounds to approximate the difference between local (0,1-hop) and global gradients.

H.1.1 Gradient of Centralized GCN (Global Gradient)

In centralized training, for a graph $\mathcal{G}$ with N nodes and d-dim feature for each node. It can be also represented as the adjacency matrix $\boldsymbol{A}$ and the feature matrix $\boldsymbol{X}$. We then consider a 1-layer graph convolutional network with model parameter $\boldsymbol{W}$ and softmax activation ϕ, which has the following form

$$\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{X}\boldsymbol{W}. \tag{7}$$

We then pass it to the softmax activation

$$\boldsymbol{Q} = \phi(\boldsymbol{Z}), \tag{8}$$

where

$$\boldsymbol{Q}_{ic} = \frac{e^{\boldsymbol{Z}_{ic}}}{\sum_{c=1}^{C} e^{\boldsymbol{Z}_{ic}}}. \tag{9}$$

$\boldsymbol{Q}_{ic}$ is then the model prediction result for node i with specific class c.

Let $f(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{W}, \boldsymbol{Y})$ represent the output of the cross-entropy loss, we have

$$f(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{W}, \boldsymbol{Y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} \boldsymbol{Y}_{ic} \log \boldsymbol{Q}_{ic}. \tag{10}$$

Equation 1 Gradient to the input of softmax layer $\frac{\partial f}{\partial \boldsymbol{Z}} = \frac{1}{N}(\boldsymbol{Q} - \boldsymbol{Y})$

Proof. At first, we calculate the gradient of f given the element $\boldsymbol{Z}_{ic}$ of the matrix $\boldsymbol{Z}$, $\frac{\partial f}{\partial \boldsymbol{Z}_{ic}}$,

$$\begin{aligned}
\frac{\partial f}{\partial \boldsymbol{Z}_{ic}} &= \frac{\partial(-\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} \boldsymbol{Y}_{ic}\log \boldsymbol{Q}_{ic})}{\partial \boldsymbol{Z}_{ic}} \\
&= \frac{\partial(-\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} \boldsymbol{Y}_{ic}\log \frac{e^{\boldsymbol{Z}_{ic}}}{\sum_{d=1}^{C} e^{\boldsymbol{Z}_{id}}})}{\partial \boldsymbol{Z}_{ic}} \\
&= \frac{\partial(-\frac{1}{N}\sum_{c=1}^{C} \boldsymbol{Y}_{ic}\log \frac{e^{\boldsymbol{Z}_{ic}}}{\sum_{d=1}^{C} e^{\boldsymbol{Z}_{id}}})}{\partial \boldsymbol{Z}_{ic}} \\
&= -\frac{1}{N}\frac{\partial(\sum_{c=1}^{C} \boldsymbol{Y}_{ic}\log \frac{e^{\boldsymbol{Z}_{ic}}}{\sum_{d=1}^{C} e^{\boldsymbol{Z}_{id}}})}{\partial \boldsymbol{Z}_{ic}} \\
&= -\frac{1}{N}\frac{\partial(\sum_{c=1}^{C}(\boldsymbol{Y}_{ic}\boldsymbol{Z}_{ic} - \boldsymbol{Y}_{ic}\log\sum_{d=1}^{C} e^{\boldsymbol{Z}_{id}}))}{\partial \boldsymbol{Z}_{ic}} \\
&= -\frac{1}{N}(\boldsymbol{Y}_{ic} - \frac{\partial(\sum_{c=1}^{C}(\boldsymbol{Y}_{ic}\log\sum_{d=1}^{C} e^{\boldsymbol{Z}_{id}}))}{\partial \boldsymbol{Z}_{ic}}) \\
&= -\frac{1}{N}(\boldsymbol{Y}_{ic} - \frac{\partial(\log\sum_{d=1}^{C} e^{\boldsymbol{Z}_{id}}))}{\partial \boldsymbol{Z}_{ic}}) \\
&= -\frac{1}{N}(\boldsymbol{Y}_{ic} - \frac{e^{\boldsymbol{Z}_{ic}}}{\sum_{d=1}^{C} e^{\boldsymbol{Z}_{id}}}) \\
&= \frac{1}{N}(\frac{e^{\boldsymbol{Z}_{ic}}}{\sum_{d=1}^{C} e^{\boldsymbol{Z}_{id}}} - \boldsymbol{Y}_{ic}) \\
&= \frac{1}{N}(\boldsymbol{Q}_{ic} - \boldsymbol{Y}_{ic})
\end{aligned} \tag{11}$$

Given the property of the matrix, we have

$$\frac{\partial f}{\partial \boldsymbol{Z}} = \frac{1}{N}(\boldsymbol{Q} - \boldsymbol{Y}).$$

□

Lemma 1 If $\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{X}\boldsymbol{B}$,

$$\frac{\partial f}{\partial \boldsymbol{X}} = \boldsymbol{A}^T \frac{\partial f}{\partial \boldsymbol{Z}} \boldsymbol{B}^T.$$

Equation 2 The gradient over the weights of GCN

$$\frac{\partial f}{\partial \boldsymbol{W}} = \frac{1}{N} \boldsymbol{X}^T \boldsymbol{A}^T (\phi(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}) - \boldsymbol{Y}). \tag{12}$$

Proof.

$$\begin{aligned} \frac{\partial f}{\partial \boldsymbol{W}} &= (\boldsymbol{A}\boldsymbol{X})^T \frac{\partial f}{\partial \boldsymbol{Z}} \\ &= \boldsymbol{X}^T \boldsymbol{A}^T \frac{\partial f}{\partial \boldsymbol{Z}} \\ &= \frac{1}{N} \boldsymbol{X}^T \boldsymbol{A}^T (\boldsymbol{Q} - \boldsymbol{Y}) \\ &= \frac{1}{N} \boldsymbol{X}^T \boldsymbol{A}^T (\phi(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}) - \boldsymbol{Y}) \end{aligned} \tag{13}$$

□

H.1.2 Gradients of local models with 0-hop communication

We then consider the federated setting. Let $\boldsymbol{A}^{N \times N}$ denote the adjacency matrix of all nodes and $\boldsymbol{A}_k^{N_k \times N_k}$ denotes the adjacency matrix of the nodes in client k. Let f_k represent the local loss function (without communication) of client k. Then the local gradient given model parameter $\boldsymbol{W}$ is

$$\frac{\partial f_k}{\partial \boldsymbol{W}} = \frac{1}{N_k} \boldsymbol{X}_k^T \boldsymbol{A}_k^T (\phi(\boldsymbol{A}_k \boldsymbol{X}_k \boldsymbol{W}) - \boldsymbol{Y}_k) \tag{14}$$

H.1.3 Gradients of local models with 1-hop communication

With 1-hop communication, let $\dot{\boldsymbol{A}}_k^{N_k \times |\mathcal{N}_k|}$ denotes the adjacency matrix of the nodes in client k and their 1-hop neighbors ($\mathcal{N}_k$ also includes the current nodes). The output of GCN with 1-hop communication (recovering 1-hop neighbor information) is

$$\phi(\dot{\boldsymbol{A}}_k \dot{\boldsymbol{X}}_k \boldsymbol{W}). \tag{15}$$

The local gradient with 1-hop communication given model parameter $\boldsymbol{W}$ is then

$$\frac{\partial \dot{f}_k}{\partial \boldsymbol{W}} = \frac{1}{N_k} \dot{\boldsymbol{X}}_k^T \dot{\boldsymbol{A}}_k^T (\phi(\dot{\boldsymbol{A}}_k \dot{\boldsymbol{X}}_k \boldsymbol{W}) - \boldsymbol{Y}_k) \tag{16}$$

H.1.4 Bound the difference of local gradient and global gradient

Assuming each client has an equal number of nodes, we have $N_k = \frac{N}{K}$. The local gradient of 0-hop communication is then

$$\frac{\partial f_k}{\partial \boldsymbol{W}} = \frac{K}{N} \boldsymbol{X}_k^T \boldsymbol{A}_k^T (\phi(\boldsymbol{A}_k \boldsymbol{X}_k \boldsymbol{W}) - \boldsymbol{Y}_k) \tag{17}$$

The difference between local gradient (0-hop) and global gradient is then

$$\begin{aligned}
&\|\frac{\partial f_k}{\partial \boldsymbol{W}} - \frac{\partial f}{\partial \boldsymbol{W}}\| \\
&= \|\frac{K}{N}\boldsymbol{X}_k^T\boldsymbol{A}_k^T(\phi(\boldsymbol{A}_k\boldsymbol{X}_k\boldsymbol{W}) - \boldsymbol{Y}_k) - \frac{1}{N}\boldsymbol{X}^T\boldsymbol{A}^T(\phi(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}) - \boldsymbol{Y})\| \\
&= \frac{1}{N}\|K\boldsymbol{X}_k^T\boldsymbol{A}_k^T(\phi(\boldsymbol{A}_k\boldsymbol{X}_k\boldsymbol{W}) - \boldsymbol{Y}_k) - \boldsymbol{X}^T\boldsymbol{A}^T(\phi(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}) - \boldsymbol{Y})\| \\
&= \frac{1}{N}\|K\boldsymbol{X}_k^T\boldsymbol{A}_k^T\phi(\boldsymbol{A}_k\boldsymbol{X}_k\boldsymbol{W}) - K\boldsymbol{X}_k^T\boldsymbol{A}_k^T\boldsymbol{Y}_k - \boldsymbol{X}^T\boldsymbol{A}^T\phi(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}) + \boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{Y}\| \\
&\leq \frac{1}{N}(\|K\boldsymbol{X}_k^T\boldsymbol{A}_k^T\phi(\boldsymbol{A}_k\boldsymbol{X}_k\boldsymbol{W}) - \boldsymbol{X}^T\boldsymbol{A}^T\phi(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W})\| + \|\boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{Y} - K\boldsymbol{X}_k^T\boldsymbol{A}_k^T\boldsymbol{Y}_k\|) \\
&= \frac{1}{N}(\|K\boldsymbol{X}_k^T\boldsymbol{A}_k^T\phi(\boldsymbol{A}_k\boldsymbol{X}_k\boldsymbol{W}) - \boldsymbol{X}^T\boldsymbol{A}^T\phi(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W})\| + \|K\boldsymbol{X}_k^T\boldsymbol{A}_k^T\boldsymbol{Y}_k - \boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{Y}\|) \\
&\lesssim \|K\boldsymbol{X}_k^T\boldsymbol{A}_k^T\phi(\boldsymbol{A}_k\boldsymbol{X}_k\boldsymbol{W}) - \boldsymbol{X}^T\boldsymbol{A}^T\phi(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W})\| + \|K\boldsymbol{X}_k^T\boldsymbol{A}_k^T\boldsymbol{Y}_k - \boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{Y}\|
\end{aligned} \tag{18}$$

Since model training is to make the model output $\phi(\boldsymbol{AXW})$ close to label matrix $\boldsymbol{Y}$, we provide an upper bound

$$\|K\boldsymbol{X}_k^T\boldsymbol{A}_k^T\boldsymbol{Y}_k - \boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{Y}\| \lesssim \|K\boldsymbol{X}_k^T\boldsymbol{A}_k^T\phi(\boldsymbol{A}_k\boldsymbol{X}_k\boldsymbol{W}) - \boldsymbol{X}^T\boldsymbol{A}^T\phi(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W})\| \tag{19}$$

Based on Equation 18 and Equation 19 , we then have

$$\|\frac{\partial f_k}{\partial \boldsymbol{W}} - \frac{\partial f}{\partial \boldsymbol{W}}\| \lesssim \|K\boldsymbol{X}_k^T\boldsymbol{A}_k^T\phi(\boldsymbol{A}_k\boldsymbol{X}_k\boldsymbol{W}) - \boldsymbol{X}^T\boldsymbol{A}^T\phi(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W})\| \tag{20}$$

By assuming the function $\boldsymbol{X}^T\boldsymbol{A}^T\phi(\boldsymbol{AXW})$ is λ-smooth w.r.t $\boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{AX}$, we have

$$\begin{aligned}
\|\frac{\partial f_k}{\partial \boldsymbol{W}} - \frac{\partial f}{\partial \boldsymbol{W}}\| &\lesssim \|K\boldsymbol{X}_k^T\boldsymbol{A}_k^T\boldsymbol{A}_k\boldsymbol{X}_k\boldsymbol{W} - \boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}\| \\
&\lesssim \|K\boldsymbol{X}_k^T\boldsymbol{A}_k^T\boldsymbol{A}_k\boldsymbol{X}_k - \boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}\|
\end{aligned} \tag{21}$$

We can then provide the following bound to compare the local gradient with 0-hop communication and the global gradient given the same model parameter $\boldsymbol{w}$

$$\|\frac{\partial f_k}{\partial \boldsymbol{w}} - \frac{\partial f}{\partial \boldsymbol{w}}\| \lesssim \|K\boldsymbol{X}_k^T\boldsymbol{A}_k^T\boldsymbol{A}_k\boldsymbol{X}_k - \boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}\|, \tag{22}$$

where $\boldsymbol{w}$ is the vectorization of model parameters $\boldsymbol{W}$.

Similarly, let $\dot{f}_k$ represent the loss function with 1-hop communication, the difference between local gradient with 1-hop communication and the global gradient is

$$\|\frac{\partial \dot{f}_k}{\partial \boldsymbol{w}} - \frac{\partial f}{\partial \boldsymbol{w}}\| \leq \lambda\|K\dot{\boldsymbol{X}}_k^T\dot{\boldsymbol{A}}_k^T\dot{\boldsymbol{A}}_k\dot{\boldsymbol{X}}_k - \boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}\|. \tag{23}$$

H.2 Convergence Analysis of 2-layer GCNs

Based on the same idea in 1-layer GCNs, we then provide the convergence analysis of 2-layer GCNs. We first derive the gradient of 2-layer GCNs in centralized, 0-hop, 1-hop and 2-hop cases. Then we provide bounds to approximate the difference between local (0, 1, 2-hop) and global gradients. Based on the Stochastic Block Model, we then be able to quantify the difference.

H.2.1 Gradient of Centralized GCN (Global Gradient)

Based on the analysis of 1-layer GCNs, for graph $\mathcal{G}$ with adjacency matrix $\boldsymbol{A}$ and feature matrix $\boldsymbol{X}$ in clients, we consider a 2-layer graph convolutional network with ReLU activation for the first layer, Softmax activation for the second layer, and cross-entropy loss, which has the following form

$$\boldsymbol{Z} = \boldsymbol{A}\phi_1(\boldsymbol{AXW}_1)\boldsymbol{W}_2, \tag{24}$$

$$\boldsymbol{Q} = \phi_2(\boldsymbol{Z}), \tag{25}$$

where

$$\boldsymbol{Q}_{ic} = \frac{e^{\boldsymbol{Z}_{ic}}}{\sum_{d=1}^{C} e^{\boldsymbol{Z}_{id}}} \tag{26}$$

The objective function is

$$f(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{Y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} \boldsymbol{Y}_{ic} \log \boldsymbol{Q}_{ic}. \tag{27}$$

We then show how to calculate the gradient $\nabla f(\mathbf{w}) = [\frac{\partial f}{\partial \boldsymbol{W}_1}, \frac{\partial f}{\partial \boldsymbol{W}_2}]$.

Equation 1 $\frac{\partial f}{\partial \boldsymbol{Z}} = \frac{1}{N}(\boldsymbol{Q} - \boldsymbol{Y})$

Equation 2 The gradient over the weights of the second layer

$$\frac{\partial f}{\partial \boldsymbol{W}_2} = \frac{1}{N}(\phi_1(\boldsymbol{W}_1^T \boldsymbol{X}^T \boldsymbol{A}^T))\boldsymbol{A}^T(\phi_2(\boldsymbol{A}\phi_1(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{W}_2) - \boldsymbol{Y}) \tag{28}$$

Proof.

$$\begin{aligned}
\frac{\partial f}{\partial \boldsymbol{W}_2} &= (\boldsymbol{A}\phi_1(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_1))^T \frac{\partial f}{\partial \boldsymbol{Z}} \\
&= (\phi_1(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_1))^T \boldsymbol{A}^T \frac{\partial f}{\partial \boldsymbol{Z}} \\
&= \frac{1}{N}(\phi_1(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_1))^T \boldsymbol{A}^T (\boldsymbol{Q} - \boldsymbol{Y}) \\
&= \frac{1}{N}(\phi_1(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_1))^T \boldsymbol{A}^T (\phi_2(\boldsymbol{A}\phi_1(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{W}_2) - \boldsymbol{Y}) \\
&= \frac{1}{N}(\phi_1(\boldsymbol{W}_1^T \boldsymbol{X}^T \boldsymbol{A}^T))\boldsymbol{A}^T(\phi_2(\boldsymbol{A}\phi_1(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{W}_2) - \boldsymbol{Y})
\end{aligned} \tag{29}$$

□

Equation 3 The gradient over the weights of the first layer.

$$\frac{\partial f}{\partial \boldsymbol{W}_1} = \frac{1}{N}(\boldsymbol{A}\phi_1'(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{A}\boldsymbol{X})^T(\phi_2(\boldsymbol{A}\phi_1(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{W}_2) - \boldsymbol{Y})\boldsymbol{W}_2^T \tag{30}$$

Proof.

$$\begin{aligned}
\frac{\partial f}{\partial \boldsymbol{W}_1} &= (\boldsymbol{A}\phi_1'(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{A}\boldsymbol{X})^T \frac{\partial f}{\partial \boldsymbol{Z}} \boldsymbol{W}_2^T \\
&= (\boldsymbol{A}\phi_1'(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{A}\boldsymbol{X})^T \frac{\partial f}{\partial \boldsymbol{Z}} \boldsymbol{W}_2^T \\
&= \frac{1}{N}(\boldsymbol{A}\phi_1'(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{A}\boldsymbol{X})^T (\boldsymbol{Q} - \boldsymbol{Y})\boldsymbol{W}_2^T \\
&= \frac{1}{N}(\boldsymbol{A}\phi_1'(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{A}\boldsymbol{X})^T(\phi_2(\boldsymbol{A}\phi_1(\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{W}_2) - \boldsymbol{Y})\boldsymbol{W}_2^T
\end{aligned} \tag{31}$$

□

H.2.2 Gradient of local models (0, 1, 2-hop)

For client k with local adjacency matrix $\boldsymbol{A}_k$, let $\dot{\boldsymbol{A}}_k^{n\times|\mathcal{N}_k|}$ denotes the adjacency matrix of the current nodes with complete edge information form their 1-hop neighbors ($\mathcal{N}_k$ also includes the current nodes), and $\ddot{\boldsymbol{A}}_k^{|\mathcal{N}_k|\times|\mathcal{N}_k^2|}$ denotes the adjacency matrix of nodes with complete edge information form their 2-hop neighbors ($\mathcal{N}_k^2$ also includes the current nodes and 1-hop neighbors).

The output of GCN without communication is

$$\phi_2(\boldsymbol{A}_k\phi_1(\boldsymbol{A}_k\boldsymbol{X}_k\boldsymbol{W}_1)\boldsymbol{W}_2). \tag{32}$$

The output of GCN with 1-hop communication is

$$\phi_2(\boldsymbol{A}_k\phi_1(\dot{\boldsymbol{A}}_k\dot{\boldsymbol{X}}_k\boldsymbol{W}_1)\boldsymbol{W}_2). \tag{33}$$

The output of GCN with 2-hop communication is

$$\phi_2(\dot{\boldsymbol{A}}_k\phi_1(\ddot{\boldsymbol{A}}_k\ddot{\boldsymbol{X}}_k\boldsymbol{W}_1)\boldsymbol{W}_2). \tag{34}$$

For 2-layer GCNs, output with 2-hop communication is the same as the centralized model.

The gradient of GCNs with 2-hop communication (recover the 2-hop neighbor information) over the weights of the first layer is then

$$\frac{\partial \ddot{f}_k}{\partial \boldsymbol{W}_1} = \frac{1}{N_k}(\dot{\boldsymbol{A}}_k\phi_1'(\ddot{\boldsymbol{A}}_k\ddot{\boldsymbol{X}}_k\boldsymbol{W}_1)\ddot{\boldsymbol{A}}_k\ddot{\boldsymbol{X}}_k)^T(\phi_2(\dot{\boldsymbol{A}}_k\phi_1(\ddot{\boldsymbol{A}}_k\ddot{\boldsymbol{X}}_k\boldsymbol{W}_1)\boldsymbol{W}_2) - \boldsymbol{Y}_k)\boldsymbol{W}_2^T. \tag{35}$$

H.2.3 Bound the difference of local gradient and global gradient

Assuming each client has equal number of nodes, we have $N_k = \frac{N}{K}$. Based on the same process in 1-layer case, we can then provide the following approximations between the local model and the global model.

The difference between the local gradient without communication and the global gradient is

$$\|\frac{\partial f_k}{\partial \boldsymbol{w}} - \frac{\partial f}{\partial \boldsymbol{w}}\| \lesssim \|K\boldsymbol{X}_k^T\boldsymbol{A}_k^T\boldsymbol{A}_k^T\boldsymbol{A}_k\boldsymbol{A}_k\boldsymbol{X}_k - \boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{A}\boldsymbol{X}\|. \tag{36}$$

The difference between the local gradient with 1-hop communication and the global gradient is

$$\|\frac{\partial \dot{f}_k}{\partial \boldsymbol{w}} - \frac{\partial f}{\partial \boldsymbol{w}}\| \lesssim \|K\dot{\boldsymbol{X}}_k^T\dot{\boldsymbol{A}}_k^T\boldsymbol{A}_k^T\boldsymbol{A}_k\dot{\boldsymbol{A}}_k\dot{\boldsymbol{X}}_k - \boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{A}\boldsymbol{X}\|. \tag{37}$$

The difference between the local gradient with 2-hop communication and the global gradient is

$$\|\frac{\partial \ddot{f}_k}{\partial \boldsymbol{w}} - \frac{\partial f}{\partial \boldsymbol{w}}\| \lesssim \|K\ddot{\boldsymbol{X}}_k^T\ddot{\boldsymbol{A}}_k^T\dot{\boldsymbol{A}}_k^T\dot{\boldsymbol{A}}_k\ddot{\boldsymbol{A}}_k\ddot{\boldsymbol{X}}_k - \boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{A}\boldsymbol{X}\|. \tag{38}$$

With more communication, the local gradient gets closer to the global gradient.

H.3 Analysis on Stochastic Block Model with Node Features

To better quantify the difference, we can analyze it on generated graphs, the Stochastic Block Model.

H.3.1 Preliminaries

Assume the node feature vector $\boldsymbol{x}$ follows the Gaussian distribution with linear projection $\boldsymbol{H}$ of node label $\boldsymbol{y}$,

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{H}\boldsymbol{y}, \sigma), \tag{39}$$

we then have the expectation of the feature matrix

$$E(\boldsymbol{X}) = E(\boldsymbol{Y}\boldsymbol{H}^T). \tag{40}$$

According to the Stochastic Block Model, we have

$$E(\boldsymbol{A}) = \boldsymbol{P} = \boldsymbol{Y}\boldsymbol{B}\boldsymbol{Y}^T. \tag{41}$$

H.3.2 Quantify the gradient difference

Based on above results, the expectation of the global gradient given the label matrix $\boldsymbol{Y}$ is

$$E(\boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{A}\boldsymbol{X}|\boldsymbol{Y}) = \boldsymbol{H}\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{B}\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{B}\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{B}\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{B}\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{H}^T. \tag{42}$$

Notice that $\boldsymbol{Y}^T\boldsymbol{Y}$ is counting the number of nodes belonging to each class. Based on this observation, we can better analyze the data distribution.

For adjacency matrix without communication

$$E(\boldsymbol{A}_k) = \mathbf{Y}_k \boldsymbol{B} \mathbf{Y}_k^T. \tag{43}$$

The expectation of the former gradient given the label matrix $\boldsymbol{Y}$ is then

$$E(\boldsymbol{X}_k^T \boldsymbol{A}_k^T \boldsymbol{A}_k^T \boldsymbol{A}_k \boldsymbol{A}_k \boldsymbol{X}_k | \boldsymbol{Y}) = \boldsymbol{H}\mathbf{Y}_k^T \mathbf{Y}_k \boldsymbol{B} \mathbf{Y}_k^T \mathbf{Y}_k \boldsymbol{B} \mathbf{Y}_k^T \mathbf{Y}_k \boldsymbol{B} \mathbf{Y}_k^T \mathbf{Y}_k \boldsymbol{B} \mathbf{Y}_k^T \mathbf{Y}_k \boldsymbol{H}^T. \tag{44}$$

For adjacency matrix with 1-hop communication

$$E(\dot{\boldsymbol{A}}_k) = \mathbf{Y}_k \boldsymbol{B} \dot{\mathbf{Y}}_k^T \tag{45}$$

The expectation of the former gradient with 1-hop communication given the label matrix $\boldsymbol{Y}$ is then

$$\begin{aligned} &E(\dot{\boldsymbol{X}}_k^T \dot{\boldsymbol{A}}_k^T \boldsymbol{A}_k^T \boldsymbol{A}_k \dot{\boldsymbol{A}}_k \dot{\boldsymbol{X}}_k | \boldsymbol{Y}) \\ &\qquad = \boldsymbol{H}\dot{\mathbf{Y}}_k^T \dot{\mathbf{Y}}_k \boldsymbol{B} \mathbf{Y}_k^T \mathbf{Y}_k \boldsymbol{B} \mathbf{Y}_k^T \mathbf{Y}_k \boldsymbol{B} \mathbf{Y}_k^T \mathbf{Y}_k \boldsymbol{B} \dot{\mathbf{Y}}_k^T \dot{\mathbf{Y}}_k \boldsymbol{H}^T. \end{aligned} \tag{46}$$

For adjacency matrix with 2-hop communication

$$E(\ddot{\boldsymbol{A}}) = \dot{\boldsymbol{Y}} \boldsymbol{B} \ddot{\boldsymbol{Y}}^T. \tag{47}$$

The expectation of the former gradient with 1-hop communication given the label matrix $\boldsymbol{Y}$ is then

$$\begin{aligned} &E(\ddot{\boldsymbol{X}}_k^T \ddot{\boldsymbol{A}}_k^T \dot{\boldsymbol{A}}_k^T \dot{\boldsymbol{A}}_k \ddot{\boldsymbol{A}}_k \ddot{\boldsymbol{X}}_k | \boldsymbol{Y}) \\ &\qquad = \boldsymbol{H}\ddot{\mathbf{Y}}_k^T \ddot{\mathbf{Y}}_k \boldsymbol{B} \dot{\mathbf{Y}}_k^T \dot{\mathbf{Y}}_k \boldsymbol{B} \mathbf{Y}_k^T \mathbf{Y}_k \boldsymbol{B} \dot{\mathbf{Y}}_k^T \dot{\mathbf{Y}}_k \boldsymbol{B} \ddot{\mathbf{Y}}_k^T \ddot{\mathbf{Y}}_k \boldsymbol{H}. \end{aligned} \tag{48}$$

The difference of gradient can then be written as

$$\begin{aligned} \|\frac{\partial \ddot{f}_k}{\partial \boldsymbol{w}} - \frac{\partial f}{\partial \boldsymbol{w}}\| \leq \lambda \| K &\ddot{\mathbf{Y}}_k^T \ddot{\mathbf{Y}}_k \boldsymbol{B} \dot{\mathbf{Y}}_k^T \dot{\mathbf{Y}}_k \boldsymbol{B} \mathbf{Y}_k^T \mathbf{Y}_k \boldsymbol{B} \dot{\mathbf{Y}}_k^T \dot{\mathbf{Y}}_k \boldsymbol{B} \ddot{\mathbf{Y}}_k^T \ddot{\mathbf{Y}}_k \\ &- \boldsymbol{Y}^T \boldsymbol{Y} \boldsymbol{B} \boldsymbol{Y}^T \boldsymbol{Y} \boldsymbol{B} \boldsymbol{Y}^T \boldsymbol{Y} \boldsymbol{B} \boldsymbol{Y}^T \boldsymbol{Y} \boldsymbol{B} \boldsymbol{Y}^T \boldsymbol{Y} \| \end{aligned} \tag{49}$$

Notice that $\mathbf{Y}_k^T \mathbf{Y}_k$ is counting the number of nodes in client k belonging to each class, $\dot{\mathbf{Y}}_k^T \dot{\mathbf{Y}}_k$ and $\ddot{\mathbf{Y}}_k^T \ddot{\mathbf{Y}}_k$ are respectively counting the number of 1-hop and 2-hop neighbors of nodes in client k belonging to each class. It can be decomposed as

$$\mathbf{Y}_k^T \mathbf{Y}_k = N_k \boldsymbol{p}_k, \tag{50}$$

We then have

$$\begin{aligned} \|\frac{\partial \ddot{f}_k}{\partial \boldsymbol{w}} - \frac{\partial f}{\partial \boldsymbol{w}}\| &\lesssim \|K \mathcal{N}_{\mathcal{V}_k}^2 \boldsymbol{p}_k \mathcal{N}_{\mathcal{V}_k}^1 \boldsymbol{p}_k N_k \boldsymbol{p}_k \mathcal{N}_{\mathcal{V}_k}^1 \boldsymbol{p}_k \mathcal{N}_{\mathcal{V}_k}^2 \boldsymbol{p}_k - N^5 \boldsymbol{p}^5 \| \|\boldsymbol{B}^4\| \\ &\lesssim \|K N_k (\mathcal{N}_{\mathcal{V}_k}^1)^2 (\mathcal{N}_{\mathcal{V}_k}^2)^2 (\boldsymbol{p}_k)^5 - N^5 \boldsymbol{p}^5 \| \\ &\lesssim \|(K N_k (\mathcal{N}_{\mathcal{V}_k}^1)^2 (\mathcal{N}_{\mathcal{V}_k}^2)^2 - N^5)(\boldsymbol{p}_k)^5\| + N^5 \|(\boldsymbol{p}_k)^5 - \boldsymbol{p}^5\| \end{aligned} \tag{51}$$

$(K N_k (\mathcal{N}_{\mathcal{V}_k}^1)^2 (\mathcal{N}_{\mathcal{V}_k}^2)^2 - N^5)$ evaluates the difference between the number of nodes with communication in local client and the number of nodes in total. $\|(\boldsymbol{p}_k)^5 - \boldsymbol{p}^5\|$ evaluates the difference between local distribution and global distribution. The second term can be bouned by

$$N^5 \|(\boldsymbol{p}_k)^5 - \boldsymbol{p}^5\| \leq N^5 (1 - \frac{1}{C})^{\frac{5}{2}} (1-p)^5. \tag{52}$$

We then work on bounding the first term.

H.4 Number of 1-hop and 2-hop neighbors for clients

We need to get the number of 1-hop neighbors $\mathcal{N}^1_{\mathcal{V}_k}$ and 2-hop neighbors $\mathcal{N}^2_{\mathcal{V}_k}$ in both i.i.d and non-i.i.d cases.

H.4.1 Number of 1-hop and 2-hop neighbors in i.i.d

For node i in other clients, the probability that it has at least one connection with the nodes in client i

$$1-(1-\alpha)^{\frac{N}{CK}}(1-\mu\alpha)^{\frac{(C-1)N}{CK}} \tag{53}$$

The expectation of 1-hop neighbor (including nodes in local client)

$$\begin{aligned}\frac{N}{K}+\frac{K-1}{K}N(1-(1-\alpha)^{\frac{N}{CK}}(1-\mu\alpha)^{\frac{(C-1)N}{CK}}) &\approx \frac{N}{K}+\frac{K-1}{K}N(1-(1-\alpha\frac{N}{CK})(1-\mu\alpha\frac{(C-1)N}{CK}))\\ &\approx \frac{N}{K}+\frac{K-1}{K}N(1-(1-\alpha\frac{N}{CK}-\mu\alpha\frac{(C-1)N}{CK}))\\ &= \frac{N}{K}+\frac{K-1}{K}N(\alpha\frac{N}{CK}+\mu\alpha\frac{(C-1)N}{CK})\\ &= \frac{N}{K}(1+(K-1)(\alpha\frac{N}{CK}+\mu\alpha\frac{(C-1)N}{CK}))\end{aligned} \tag{54}$$

Notice that it is $(1+(K-1)(\alpha\frac{N}{CK}+\mu\alpha\frac{(C-1)N}{CK})$ times the number of local nodes.

Similarly, approximated expectation of 2-hop neighbor (including nodes in local client). This approximation is provided based on that in expectation there is no label distribution shift between 2-hop nodes and 1-hop nodes.

$$\frac{N}{K}(1+(K-1)(\alpha\frac{N}{CK}+\mu\alpha\frac{(C-1)N}{CK}))^2 \tag{55}$$

H.4.2 Number of 1-hop and 2-hop neighbors in non-i.i.d.

Expectation of 1-hop neighbor (including nodes in local client)

$$\begin{aligned}&\frac{N}{K}+\frac{K-1}{K}N(1-(1-\mu\alpha)^{\frac{N}{K}})\\ &\approx \frac{N}{K}(1+\mu\alpha\frac{K-1}{K}N)\end{aligned} \tag{56}$$

Approximated expectation of 2-hop neighbor (Including nodes in local client).

$$\frac{N}{K}(1+\mu\alpha\frac{K-1}{K}N)^2 \tag{57}$$

H.4.3 Number of 1-hop and 2-hop neighbors in non-i.i.d.

Expectation of 1-hop neighbor (including nodes in local client)

$$\begin{aligned}\frac{N}{K}+\frac{K-1}{K}N(1-(1-\alpha)^{\frac{Np}{CK}}(1-\mu\alpha)^{\frac{N(C-p)}{CK}}) &\approx \frac{N}{K}+\frac{K-1}{K}N(\alpha\frac{N}{CK}((1-\mu)p+\mu C))\\ &= \frac{N}{K}+\frac{K-1}{K}N(\alpha\frac{N}{CK}(1-\mu)p+\alpha\frac{N}{CK}\mu C)\\ &= \frac{N}{K}(1+(K-1)(\alpha\frac{N}{CK}(1-\mu)p+\mu\alpha\frac{N}{K}))\end{aligned} \tag{58}$$

Approximated expectation of 2-hop neighbor (including nodes in local client)

$$\frac{N}{K}(1+(K-1)(\alpha\frac{N}{CK}(1-\mu)p+\mu\alpha\frac{N}{K}))^2 \tag{59}$$

H.5 Data Distribution with Labels

We assume each label have the same number of nodes. Each client k have the same number of nodes $N_k = \frac{N}{K}$.

For global label distribution, we have

$$\boldsymbol{p} = diag(\frac{1}{C}, ..., \frac{1}{C}) \tag{60}$$

H.5.1 i.i.d

The local label distribution is the same as the global distribution in i.i.d condition.

$$\boldsymbol{p}_k = diag(\frac{1}{C}, ..., \frac{1}{C}) \tag{61}$$

For local gradient without communication and global gradient,

$$\begin{aligned}
\|\frac{\partial f_k}{\partial \boldsymbol{w}} - \frac{\partial f}{\partial \boldsymbol{w}}\| &\lesssim \|(K(N_k)^5 - N^5)diag(\frac{1}{C}, ..., \frac{1}{C})^5 \boldsymbol{B}^4\| \\
&\lesssim (1 - K\frac{(N_k)^5}{N^5})N^5 \|diag(\frac{1}{C}, ..., \frac{1}{C})^5 \boldsymbol{B}^4\| \\
&\lesssim (1 - K\frac{(N_k)^5}{N^5})\frac{N^5}{C^5}\|\boldsymbol{B}^4\| \\
&\lesssim (1 - \frac{1}{K^4})\frac{N^5}{C^5}\|\boldsymbol{B}^4\|
\end{aligned} \tag{62}$$

For local gradient with 1-hop communication and global gradient,

$$\begin{aligned}
\|\frac{\partial \dot{f}_k}{\partial \boldsymbol{w}} - \frac{\partial f}{\partial \boldsymbol{w}}\| &\lesssim \|(K(N_k)^3|\mathcal{N}_k|^2 - N^5)diag(\frac{1}{C}, ..., \frac{1}{C})^5 \boldsymbol{B}^4\| \\
&\lesssim \|(K\frac{N^3}{K^3}(\frac{N}{C} + \frac{C-1}{C}N(\alpha\frac{N}{C} + \mu\alpha\frac{(C-1)N}{CK}))^2 - N^5)diag(\frac{1}{C}, ..., \frac{1}{C})^5 \boldsymbol{B}^4\| \\
&\lesssim \|(C\frac{N^5}{C^5}(1 + (C-1)(\alpha\frac{N}{C} + \mu\alpha\frac{(C-1)N}{CK}))^2 - N^5)diag(\frac{1}{C}, ..., \frac{1}{C})^5 \boldsymbol{B}^4\| \\
&\lesssim (1 - \frac{1}{C^4}(1 + (C-1)(\alpha\frac{N}{C} + \mu\alpha\frac{(C-1)N}{CK}))^2)\frac{N^5}{C^5}\|\boldsymbol{B}^4\|
\end{aligned} \tag{63}$$

For local gradient with 2-hop communication and global gradient,

$$\begin{aligned}
\|\frac{\partial \ddot{f}_k}{\partial \boldsymbol{w}} - \frac{\partial f}{\partial \boldsymbol{w}}\| &\lesssim \|(K(N_k)|\mathcal{N}_k|^2|\mathcal{N}_k^2|^2 - N^5)diag(\frac{1}{C}, ..., \frac{1}{C})^5 \boldsymbol{B}^4\| \\
&\lesssim \|(K\frac{N}{C}(\frac{N}{C} + \frac{C-1}{C}N(\alpha\frac{N}{C} + \mu\alpha\frac{(C-1)N}{CK}))^2 \\
&\quad ((\frac{N}{C} + \frac{C-1}{C}N(\alpha\frac{N}{C} + \mu\alpha\frac{(C-1)N}{CK}))^2)^2 - N^5)diag(\frac{1}{C}, ..., \frac{1}{C})^5 \boldsymbol{B}^4\| \\
&\lesssim \|(K\frac{N^5}{C^5}(1 + (C-1)(\alpha\frac{N}{C} + \mu\alpha\frac{(C-1)N}{CK}))^6 - N^5)diag(\frac{1}{C}, ..., \frac{1}{C})^5 \boldsymbol{B}^4\| \\
&\lesssim (1 - \frac{1}{C^4}(1 + (C-1)(\alpha\frac{N}{C} + \mu\alpha\frac{(C-1)N}{CK}))^6)\frac{N^5}{C^5}\|\boldsymbol{B}^4\|
\end{aligned} \tag{64}$$

For non-i.i.d, we can simply replace the number of 1-hop and 2-hop neighbors.

I Communication Cost under SBM

Assume the number of clients K is equal to the number of labels types in the graph G. Table 5 shows the communication cost of FedGCN and BDS-GCN Wan et al. (2022). Distributed training methods like BDS-GCN requires communication per local update, which makes the communication cost increase linearly with the number of global training round T and number of local updates E. FedGCN only requires low communication cost at the initial step.

Methods	1-hop	L-hop	BDS-GCN
Generic Graph	$C_1 + Nd$	$C_1 + \sum_{k=1}^{K} \lvert\mathcal{N}_k^{L-1}\rvert d$	$LTE\rho d \sum_{k=1}^{K} \lvert\mathcal{N}_k^1 / \mathcal{V}_k\rvert$

Table 5: Communication costs of FedGCN and BDS-GCN on generic graph. BDS-GCN requires communication at every local updates.

I.1 Server Aggregation

We consider communication cost of node i in client $c(i)$. For node i, the server needs to receive messages from $c(i)$ (note that $c(i)$ needs send the local neighbor aggregation) and other clients containing the neighbors of node i .

I.1.1 Non-i.i.d.

Possibility that there is no connected node in client j for node i is

$$(1-\mu\alpha)^{\frac{N}{K}}. \tag{65}$$

Possibility that there is at least one connected node in client j for node i is

$$1-(1-\mu\alpha)^{\frac{N}{K}}. \tag{66}$$

Number of clients that node i needs to communicate with is

$$1+(K-1)(1-(1-\mu\alpha)^{\frac{N}{K}}). \tag{67}$$

The communication cost of N nodes is

$$N(1+(K-1)(1-(1-\mu\alpha)^{\frac{N}{K}}))d. \tag{68}$$

1-order Approximation To better understanding the communication cost, we can expand the form to provide 1-order approximation

$$(1-\mu\alpha)^{\frac{N}{K}} \approx 1-\mu\alpha\frac{N}{K} \tag{69}$$

Possibility that there is no connected node in client j for node i is

$$1-(1-\mu\alpha)^{\frac{N}{K}} \approx 1-1+\mu\alpha\frac{N}{K} = \mu\alpha\frac{N}{K}. \tag{70}$$

The number of clients that node i needs to communicate with is then

$$1+(K-1)(1-(1-\mu\alpha)^{\frac{N}{K}}) \approx 1+(K-1)\mu\alpha\frac{N}{K}. \tag{71}$$

I.1.2 i.i.d.

Possibility that there is no connected node in client j for node i is

$$(1-\alpha)^{\frac{N}{CK}}(1-\mu\alpha)^{\frac{(C-1)N}{CK}}. \tag{72}$$

Possibility that there is at least one connected node in client j for node i is

$$1-(1-\alpha)^{\frac{N}{CK}}(1-\mu\alpha)^{\frac{(C-1)N}{CK}}. \tag{73}$$

Number of clients that node i needs to communicate with is

$$1+(C-1)(1-(1-\alpha)^{\frac{N}{CK}}(1-\mu\alpha)^{\frac{(C-1)N}{CK}}). \tag{74}$$

Node i needs to communicate with more clients in i.i.d. than the case in non-i.i.d.

The communication cost of N nodes is

$$N(1+(C-1)(1-(1-\alpha)^{\frac{N}{CK}}(1-\mu\alpha)^{\frac{(C-1)N}{CK}}))d. \tag{75}$$

1-order Approximation

The number of clients that node i needs to communicate with is then

$$\begin{aligned}
1-(1-\alpha)^{\frac{N}{CK}}(1-\mu\alpha)^{\frac{(C-1)N}{CK}} &\approx 1-(1-\alpha\frac{N}{CK})(1-\mu\alpha\frac{(C-1)N}{CK}) \\
&= 1-(1-\alpha\frac{N}{CK}-\mu\alpha\frac{(C-1)N}{CK}+\alpha\frac{N}{CK}\mu\alpha\frac{(C-1)N}{CK}) \\
&= \alpha\frac{N}{CK}+\mu\alpha\frac{(C-1)N}{CK}-\alpha\frac{N}{CK}\mu\alpha\frac{(C-1)N}{CK} \\
&\approx \alpha\frac{N}{CK}+\mu\alpha\frac{(C-1)N}{CK}.
\end{aligned} \tag{76}$$

The number of clients that node i needs to communicate with is then

$$(1+(C-1)(\alpha\frac{N}{CK}+\mu\alpha\frac{(C-1)N}{CK})). \tag{77}$$

I.1.3 Non-i.i.d.

Similarly, let $\boldsymbol{p}$ denote the percent of i.i.d., we then have the communication cost

$$N(1+(C-1)(1-(1-\alpha)^{\frac{Np}{CK}}(1-\mu\alpha)^{\frac{N(C-p)}{CK}}))d. \tag{78}$$

1-order Approximation

The number of clients that node i needs to communicate with is then

$$\begin{aligned}
1-(1-\alpha)^{\frac{Np}{CK}}(1-\mu\alpha)^{\frac{N(C-p)}{CK}} &\approx 1-(1-\alpha\frac{Np}{CK})(1-\mu\alpha\frac{N(C-p)}{CK}) \\
&= 1-(1-\alpha\frac{Np}{CK}-\mu\alpha\frac{N(C-p)}{CK}+\alpha\frac{Np}{CK}\mu\alpha\frac{N(C-p)}{CK} \\
&= \alpha\frac{Np}{CK}+\mu\alpha\frac{N(C-p)}{CK}-\alpha\frac{Np}{CK}\mu\alpha\frac{N(C-p)}{CK} \\
&\approx \alpha\frac{Np}{CK}+\mu\alpha\frac{N(C-p)}{CK} \\
&= \alpha\frac{N}{CK}(p+\mu(C-p)) \\
&= \alpha\frac{N}{CK}(p-\mu p+\mu C) \\
&= \alpha\frac{N}{CK}((1-\mu)p+\mu C).
\end{aligned} \tag{79}$$

The communication cost of all nodes is then

$$
\begin{aligned}
N(1+(C-1)\alpha\frac{N}{CK}((1-\mu)p+\mu C))d. &= (((1-\mu)p+\mu C)\frac{\alpha N(C-1)}{CK}+1)Nd. \\
&= (((1-\mu)p+\mu C)\frac{\alpha N(C-1)}{CK}+1)Nd. \\
&= (\frac{(1-\mu)\alpha N(C-1)}{CK}p+\frac{\mu\alpha N(C-1)}{C}+1)Nd.
\end{aligned} \tag{80}
$$

I.2 Server sends to clients

Since the aggregations of neighbor features have been calculated in the server, it then needs to send the aggregations back to clients.

For 1-hop communication, each client requires the aggregations of neighbors (1-hop) of its local nodes, which equals to the number of local nodes times the size of the node feature,

$$
\sum_{k=1}^{K}|\mathcal{V}_k|d = Nd. \tag{81}
$$

For 2-hop communication, each client requires the aggregations of 2-hop neighbors of its local nodes, which equals to the number of 1-hop neighbors times the size of the node feature,

$$
\sum_{k=1}^{K}|\mathcal{N}_{\mathcal{V}_k}|d \tag{82}
$$

The number of neighbors in partial i.i.id for client k

$$
\begin{aligned}
\frac{N}{C}+\frac{C-1}{C}N(1-(1-\alpha)^{\frac{Np}{CK}}(1-\mu\alpha)^{\frac{N(C-p)}{CK}}) &\approx \frac{N}{C}+\frac{C-1}{C}N(\alpha\frac{N}{CK}((1-\mu)p+\mu C)) \\
&= \frac{N}{C}+\frac{C-1}{C}N(\alpha\frac{N}{CK}(1-\mu)p+\alpha\frac{N}{C}\mu)
\end{aligned} \tag{83}
$$

Then the number of neighbors in partial i.i.id for for all clients

$$
N+(C-1)N(1-(1-\alpha)^{\frac{Np}{CK}}(1-\mu\alpha)^{\frac{N(C-p)}{CK}}) \approx N+(C-1)N(\alpha\frac{N}{CK}(1-\mu)p+\alpha\frac{N}{C}\mu) \tag{84}
$$

The communication cost is then

$$
\begin{aligned}
&(N+(C-1)N(\alpha\frac{N}{CK}(1-\mu)p+\alpha\frac{N}{CK}\mu C))d = (1+(C-1)(\alpha\frac{N}{CK}(1-\mu)p+\alpha\frac{N}{C}\mu))Nd \\
&= (1+(C-1)\alpha\frac{N}{CK}(1-\mu)p+\mu\alpha(C-1)\frac{N}{C})Nd
\end{aligned} \tag{85}
$$

For L-hop communication, each client requires the aggregations of L-hop neighbors of its local nodes, which equals to the number of $(L-1)$-hop neighbors times the size of the node feature,

$$
\sum_{k=1}^{K}|\mathcal{N}_{\mathcal{V}_k}^{L-1}|d. \tag{86}
$$

J Negative Social Impacts of the Work

We believe that our work overall may have a positive social impact, as it helps to protect user privacy during federated training of GCNs for node-level prediction problems. However, by enabling such training to occur without compromising privacy, there is a chance that we could enable improved training of models with negative social impact. For example, models might more accurately classify users in social networks due to their ability to leverage a larger, cross-client dataset of users in the training. Depending on the model being trained, these results could be used against such users, e.g., targeting dissidents under an authoritarian regime. We believe that such negative impacts are no more likely than positive impacts from improved training, e.g., allowing an advertising company to send better products to users through improved predictions of what they will like. This work itself is agnostic to the specific machine learning model being trained.

References

Aleksandar Bojchevski, Johannes Gasteiger, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling graph neural networks with approximate pagerank. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2464–2473.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In International conference on the theory and application of cryptology and information security. Springer, 409–437.

Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. 2020. Sign: Scalable inception graph neural networks. arXiv preprint arXiv:2004.11198 (2020).

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. Advances in neural information processing systems 33 (2020), 22118–22133.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016).

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics. PMLR, 1273–1282.

PALISADE. 2020. PALISADE Release. `https://gitlab.com/palisade/palisade-release`

Cheng Wan, Youjie Li, Ang Li, Nam Sung Kim, and Yingyan Lin. 2022. BNS-GCN: Efficient full-graph training of graph convolutional networks with partition-parallelism and random boundary node sampling. Proceedings of Machine Learning and Systems 4 (2022), 673–693.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In International conference on machine learning. PMLR, 6861–6871.