FedGCN: Convergence and Communication Tradeoffs in Federated Training of Graph Convolutional Networks

Yuhang Yao 1 Weizhao Jin 2 Srivatsan Ravi 2 Carlee Joe-Wong 1

Abstract

Methods for training models on graphs distributed across multiple clients have recently grown in popularity, due to the size of these graphs as well as regulations on keeping data where it is generated, like GDPR in the EU. However, a single connected graph cannot be disjointly partitioned onto multiple distributed clients due to the cross-client edges connecting graph nodes. Thus, distributed methods for training a model on a single graph incur either significant communication overhead between clients or a loss of available information to the training. We introduce the Federated Graph Convolutional Network (FedGCN) algorithm, which uses federated learning to train GCN models for semi-supervised node classification on large graphs with fast convergence and little communication. Compared to prior methods that require communication among clients at each training round, FedGCN clients only communicate with the central server in one pre-training step, greatly reducing communication costs. We theoretically analyze the tradeoff between FedGCN’s convergence rate and communication cost under different data distributions and introduce a general framework that can be used for analysis of all edge-completion-based GCN training algorithms. Experimental results show that our FedGCN algorithm achieves 51.7% faster convergence on average and at least 100× less communication cost compared to prior work.

1 Introduction

Graph convolutional networks (GCNs) have been widely used for applications ranging from fake news detection in social networks to anomaly detection in sensor networks (Benamira et al., 2019; Zhang et al., 2020). This data, however, can be too large to store at a single server, e.g., records of billions of users’ website visits. Strict data protection regulations such as General Data Protection Regulation (GDPR) in Europe (EU) and Payment Aggregators and Payment Gateways (PAPG) in India (IN) also require the private data only stored in local clients. In non-graph settings, federated learning has recently shown promise for training models on data that is kept at multiple clients (Zhao et al., 2018; Yang et al., 2021). Some papers have proposed federated training of GCNs (He et al., 2021a; Zhang et al., 2021). Typically, these consider a framework in which each client has access to a subset of a large graph, and clients iteratively compute local updates to a semi-supervised model on their local subgraphs, which are occasionally aggregated at a central server, as in Figure 1’s illustration of the federated node classification task to predict the unknown labels of local nodes in each client.

The main challenge of applying federated learning to GCN training tasks involving a single large graph is that disjoint partitions across clients are not possible. In Figure 1, for example, we see that when nodes are partitioned among clients, some edges will cross different clients. We refer to these as “cross-client edges”. However, GCNs require information about a node’s neighbors to be aggregated in order to construct an embedding of each node that is used to accomplish tasks such as node classification and link prediction. Many federated graph training algorithms (Wang et al., 2020a; He et al., 2021b) simply ignore the information from neighbors located at another client, which may result in less accurate models due to loss of information. Sending the features of neighboring nodes to other clients, however, can introduce significant communication overhead and reveal private node information to other clients (Wan et al., 2022).

Prior works on federated or distributed graph training reduce cross-client information loss by communicating information about nodes’ neighbors at other clients in each training round (Scardapane et al., 2020; Wan et al., 2022; Zhang et al., 2021). We instead realize that the information needed to train a GCN only needs to be communicated once, before training. Moreover, each node at a given client only needs to know the accumulated information about that node’s neighbors at each other client, allowing us to communicate only the accumulated information. In practice, there may be many more nodes than clients, so each client would receive...
FedGCN: Convergence and Communication Tradeoffs in Federated Training of Graph Convolutional Networks

Figure 1. Federated GCN training schematic for node classification, with colors indicating the node labels. Nodes in a graph (shown as circles) are partitioned across clients, and dashed lines show cross-client edges between nodes at different clients. Arrows indicate that each client can exchange updates with a central server during the training process. The task is to predict the unknown labels of the grey nodes in each client. Other nodes’ labels are known.

information accumulated over multiple nodes. For example, clients might represent a company’s supply chain or social network data in different countries, which cannot leave the country due to privacy regulations. Each client would then receive aggregated information about all of a node’s neighbors in a different country. We propose the FedGCN algorithm for distributed GCN training based on these insights. FedGCN greatly reduces communication costs and speeds up convergence without information loss, compared with existing distributed settings that reduce cross-client information loss with communication in each training round (Scardapane et al., 2020; Wan et al., 2022; Zhang et al., 2021)

Figure 2. Cross-client graph with i.i.d (left) and non-i.i.d. (right) data distribution. Node color represents the class of the node. Data distribution affects the number of cross-client edges.

In some settings, we can further reduce FedGCN’s required communication without compromising the trained model’s accuracy. In particular, GCN models for node classification rely on the fact that nodes of the same class will have more edges connecting them, as shown in Figure 2. If nodes in each client have balanced types of classes (independent and identically distributed, or i.i.d., data), there will be many edges across clients and FedGCN may still require much communication. If these nodes are concentrated at the same clients (a version of the non-i.i.d. data often considered in federated learning), then ignoring cross-client edges discards little information, and FedGCN’s communication round may be unnecessary. The model, however, may not converge, as federated learning may converge poorly when client data is non-i.i.d. (Zhao et al., 2018). It is thus unclear under which circumstances more communication is worth the convergence improvement.

We analytically quantify the convergence rate of FedGCN with various degrees of communication, under both i.i.d. and non-i.i.d. client data. To the best of our knowledge, we are the first to analytically illustrate the resulting tradeoff between a fast convergence rate (which intuitively requires more information from cross-client edges) and low communication cost, which we do by considering a stochastic block model (Lei & Rinaldo, 2015; Keriven et al., 2020) of the graph topology. We can thus quantify scenarios in which FedGCN’s communication significantly accelerates the GCN’s convergence.

Communicating data between clients additionally runs the risk of compromising privacy. FedGCN’s accumulation of node data may itself preserve privacy if multiple nodes’ information is aggregated. Unlike prior works that also face privacy challenges due to communicating node information between clients (Zhang et al., 2021), we further guarantee privacy by leveraging Fully Homomorphic Encryption (FHE) (Ni et al., 2021) to secure the neighbor feature aggregation and model gradient aggregation. In the extreme case when nodes only have one cross-client neighbor, resulting in limited privacy from accumulating node features, FedGCN can further integrate optional differential privacy techniques (Wei et al., 2020).
In summary, our work has the following contributions:

- We introduce FedGCN, an efficient framework for federated training of GCNs to solve node-level prediction tasks with limited communication and information loss, which also leverages Fully Homomorphic Encryption for enhanced privacy guarantees.
- We theoretically analyze the convergence rate and communication cost of FedGCN compared to prior methods, as well as its dependence on the data distribution. We can thus quantify the usefulness of communicating different amounts of cross-client information.
- Our experiments on both synthetic and real-world datasets demonstrate that FedGCN outperforms existing distributed GCN training methods in most cases with a fast convergence rate, higher accuracy, and orders-of-magnitude lower communication cost.

We outline related works in Section 2 before introducing the problem of node classification in graphs in Section 3. We then introduce FedGCN in Section 4 and analyze its performance theoretically (Section 5) and experimentally (Section 6) before concluding in Section 7.

2 RELATED WORK

Graph neural networks aim to learn representations of graph-structured data that capture features associated with graph nodes and edges between nodes (Bronstein et al., 2017). GCNs (Kipf & Welling, 2016), GraphSage (Hamilton et al., 2017), and Graph Attention Networks (GAT) (Veličković et al., 2017) perform well on tasks like node classification or link prediction. Several works provide a theoretical analysis of these models’ performance based on the stochastic block model (SBM) of the graph topology (Zhou & Amini, 2019; Lei & Rinaldo, 2015; Keriven et al., 2020; Keriven & Vaïter, 2020). We similarly adopt the SBM to quantify FedGCN’s performance.

Federa[ted learning] was first proposed by McMahan et al. (2017)’s widely adopted FedAvg algorithm, which allows clients to train a common model on their collective data while keeping personal data on clients. FedAvg and most of its variants use a central server to periodically combine local client updates into a new global model. However, FedAvg may not converge if data from different clients is non-i.i.d. (Zhao et al., 2018; Li et al., 2019b) and some clients do not regularly contribute updates (Yang et al., 2021), as may occur in practice with multiple distributed clients. We show similar results for federated graph training.

Federa[ted learning on graph neural networks] is a topic of recent interest (He et al., 2021a). To learn tasks for multiple graphs at different clients (e.g., graph classification (Zhang et al., 2018) and image classification (Li et al., 2019a)), GraphFL (Wang et al., 2020a) is a model-agnostic meta learning approach, while ASFGNN (Zheng et al., 2021) is a Bayesian optimization technique to automatically tune the hyper-parameters of all clients for separated graphs.

FedGCN instead considers semi-supervised tasks on a single large graph (e.g., for node classification), for which existing methods generally ignore cross-client edges (He et al., 2021a). Scardapane et al. (2020)’s distributed GNN proposes a training algorithm communicating the neighbor features and intermediate outputs of GNN layers among clients with expensive communication costs. BDS-GCN (Wan et al., 2022) then proposes to sample cross-client neighbors. These methods may violate client privacy by revealing per-node information to other clients. FedSage+ (Zhang et al., 2021) recovers missing neighbors for the input graph based on the node embedding, which requires fine-tuning a linear model of neighbor generation and may not fully recover the cross-client information. It is further vulnerable to the data reconstruction attack, compromising privacy.

All of the above works further require communication at every training round, introducing high communication overhead. FedGCN, in contrast, enables the private recovery of cross-client neighbor information with a single, pre-training communication round that utilizes FHE. We provide theoretical bounds on FedGCN’s convergence, which may also be applied to analyze these previously proposed algorithms.

3 FEDERATED SEMI-SUPERVISED NODE CLASSIFICATION

In this section, we formalize the problem of node classification on a single graph and introduce the federated setting in which we aim to solve this problem.

We consider a graph \( G = (V, E) \), where \( V = \{1, \ldots, N\} \) is the set of \( N \) nodes and \( E \) is a set of edges between them. The graph can be equivalently described by a weighted adjacency matrix \( A \in \mathbb{R}^{N \times N} \), where each entry \( A_{ij} \) indicates the weight of an edge from node \( i \) to node \( j \) (if the edge does not exist, the weight is zero). Every node \( i \in V \) is associated with a feature vector \( x_i \in \mathbb{R}^d \). Each node \( i \) in a subset \( V^{\text{train}} \subset V \) is associated with a corresponding label \( y_i \), which is used in the training process. Semi-supervised node classification aims to assign labels to nodes in the remaining set \( V \setminus V^{\text{train}} \), based on their feature vectors and connections to other nodes. We train a GCN model to do so.

GCNs (Kipf & Welling, 2016) consist of multiple convolutional layers, each of which constructs a node embedding by aggregating the features of its neighboring nodes. Typically, the node embedding matrix \( H^{(l)} \) for each layer...
$l = 1, 2, \ldots, L$ is initialized to $H^{(0)} = X$, the matrix of features for each node (i.e., each row of $X$ corresponds to the features for one node), and follows the propagation rule $H^{(l+1)} = \phi(AH^{(l)}W^{(l)})$. Here the $W^{(l)}$ are parameters to be learned, $A$ is the weighted adjacency matrix, and $\phi$ is an activation function. Typically, $\phi$ is chosen as the softmax function in the last layer, so that the output can be interpreted as the probabilities of a node lying in each class, with ReLU activations in the preceding layers. The embedding of each node $i \in V$ at layer $l+1$ is then

$$h_i^{(l+1)} = \phi \left( \sum_{j \in N_i} A_{ij}h_j^{(l)}W^{(l)} \right), \quad (1)$$

which can be computed from the previous layer’s embedding $h_j^{(l)}$ for each neighbor $j$ and the weights $A_{ij}$ on edges originating at node $i$. For a GCN with $L$ layers in this form, the output for node $i$ will depend on neighbors up to $L$ steps away (i.e., there exists a path of no more than $L$ edges to node $i$). We denote this set by $N_i^L$ (note that $i \in N_i^L$) and refer to these nodes as $L$-hop neighbors of $i$.

To solve the node classification problem in federated settings, as described in Figure 1, we consider, as usual in federated learning, a central server with $K$ clients. The graph $G = (V, E)$ is separated across the $K$ clients, each of which has a sub-graph $G_k = (V_k, E_k)$. Here $\bigcup_{k=1}^K V_k = V$ and $V_i \bigcap V_j = \emptyset$ for all $i \neq j \in [K]$, i.e., the nodes are disjointly partitioned across clients. The features of nodes in the set $V_k$ can then be represented as the matrix $X_k$. The cross-client edges of client $k$, $E_k^c$, for which the nodes connected by the edge are at different clients, are known to the client $k$. We use $V_k^{train} \subset V_k$ to denote the set of training nodes with associated labels $y_i$. The task of federated semi-supervised node classification is then to assign labels to nodes in the remaining set $V_k \setminus V_k^{train}$ for each client $k$.

Applying GCNs in the federated setting immediately raises a challenge. As seen from (1), in order to find the embedding of the $i$-th node in the $l$-th layer, we need the previous layer’s embedding $h_j^{(l)}$ for all neighbors of node $i$. In the federated setting, however, some of these neighbors may be located at other clients, and thus their embeddings must be iteratively sent to the client that contains node $i$ for each layer at every training round. He et al. (2021a) ignore these neighbors, considering only $G_k$ and $E_k$ in training the model, while Scardapane et al. (2020); Wan et al. (2022); Zhang et al. (2021) require such communication, which may lead to high overhead and privacy costs. FedGCN provides a communication-efficient method to account for these neighbors.

### 4 Federated Graph Convolutional Network

In order to overcome the challenges outlined in Section 3, we propose our Federated Graph Convolutional Network (FedGCN) algorithm. In this section, we first introduce our federated training method with communication at the initial step and then outline the corresponding training algorithm.

**Federating Graph Convolutional Networks.** In the federated learning setting, let $c(i)$ denote the index of the client that contains node $i$ and $W^{(l)}_{c(i)}$ denote the weight matrix of the $l$-th GCN layer of client $c(i)$. The embedding of node $i$ at layer $l+1$ is then

$$h_i^{(l+1)} = \phi \left( \sum_{j \in N_i} A_{ij}h_j^{(l)}W^{(l)}_{c(i)} \right). \quad (2)$$

Note that the weights $W^{(l)}_{c(i)}$ may differ from client to client, due to the local training in federated learning. In practice, GCNs often require only two or three layers for node-level prediction tasks (node classification and link prediction) (Kipf & Welling, 2016) to have sufficient performance. We experimentally validate the need for a limited number of layers in the paper’s appendix. We can then write the computation of a 2-layer federated GCN as

$$\hat{y}_i = \phi \left( \sum_{j \in N_i} A_{ij}\phi \left( \sum_{m \in N_j} A_{jm}x_m^T W^{(1)}_{c(i)} W^{(2)}_{c(i)} \right) \right). \quad (3)$$

We thus see that, to evaluate this model, it suffices for the client $k = c(i)$ to receive the following messages

$$\sum_{j \in N_i} A_{ij}x_j, \text{ and } \left\{ \sum_{m \in N_j} A_{jm}x_j \right\}_{j \in N_i \setminus /i}, \quad (4)$$

which are the feature aggregation of 1-hop and 2-hop neighbors of node $i$ respectively. Note that this information does not change over the course of the model training, as it simply depends on the (fixed) adjacency matrix $A$ and node features $x$. The client also naturally knows $\{A_{ij}\}_{\forall j \in N_i}$, which is included in $E_k \cup E_k^c$.

One way to obtain the above information is to receive the following message from clients $z$ that contain at least one two-hop neighbor of $k$:

$$\sum_{j \in N_i} \mathbb{I}_z(c(j))A_{ij}x_j, \text{ and } \forall j \in N_i, \sum_{m \in N_j} \mathbb{I}_z(c(m))A_{jm}x_m. \quad (5)$$

Here the indicator $\mathbb{I}_z(c(m))$ is 1 if $z = c(m)$ and zero otherwise. More generally, for a $L$-layer GCN, each layer
requires the following information:

\[ \forall j \in \mathcal{N}_i^L, \sum_{m \in \mathcal{N}_j} \mathbb{I}_k(e(m)) \cdot A_{jm} x_m. \quad (6) \]

Further, \( E_i^{L-1} \), i.e., the set of edges up to \( L - 1 \) hops away from node \( i \), is needed for normalization of \( A \). However, this method requires communication among multiple pairs of clients and suffers privacy leakage in a case when there is only one neighbor node in the client.

To avoid this overhead, we can instead send the aggregation of each client to the central server; the server then calculates the sum of neighbor features of node \( i \)

\[ \sum_{j \in \mathcal{N}_i} A_{ij} x_j = \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \mathbb{I}_k(e(j)) \cdot A_{ij} x_j \quad (7) \]

The server can then send the required feature aggregation in (4) back to each client \( k \). Thus, we only need to send the accumulated features of each node’s (possibly multi-hop) neighbors, in order to evaluate the GCN model. If there are multiple neighbors stored in other clients, this accumulation serves to protect their individual privacy\(^1\). For the computation of all nodes \( V_k \) stored in client \( k \) with an \( L \)-layer GCN, the client needs to receive \( \{ \sum_{j \in \mathcal{N}_i} A_{ij} x_j \}_{i \in \mathcal{N}_k^L} \), where \( \mathcal{N}_k^L \) is the set of \( L \)-hop neighbors of nodes \( V_k \).

FedGCN is based on the insight that GCNs require only the accumulated information of the \( L \)-hop neighbors of each node at each client, which may be communicated in advance of the training. In practice, however, even this limited communication may be infeasible. If \( L \) is too large, then the \( L \)-hop neighbors may actually consist of almost the entire graph (many social network graphs have diameters < 10), which might introduce prohibitive storage and communication requirements when there are many clients. Thus, we design FedGCN to accommodate three types of communication approximations, according to the most appropriate choice for a given application:

- **No communication (0-hop):** In some cases, e.g., any communication might reveal private information. In this case, each client simply trains on \( G_k \) and ignores cross-client edges, as in prior work.

- **One-hop communication:** If some communication is permissible, we may use accumulation of feature information from nodes’ one-hop neighbors, \( \{ \sum_{j \in \mathcal{N}_i} A_{ij} x_j \}_{i \in \mathcal{N}_k} \), to approximate the computation of the GCN. Since we only include one-hop neighbors of each node, this is unlikely to introduce significant memory or communication overhead as long as the graph is sufficiently sparse (as is often the case in practice, e.g., in social networks).

- **Two-hop communication:** Many GCNs used in practice for node classification only contain two layers (Kipf & Welling, 2016). Thus, communicating the information, \( \{ \sum_{j \in \mathcal{N}_i} A_{ij} x_j \}_{i \in \mathcal{N}_k} \), can perfectly recover all neighboring nodes’ information (i.e., there is no information loss). This choice requires more communication than one-hop communication as there are more two-hop neighbors than one-hop neighbors.

### Secure Neighbor Feature Aggregation

To guarantee privacy during the aggregation process of accumulated features, we leverage Fully Homomorphic Encryption (FHE) to construct a secure neighbor feature aggregation function. Fully Homomorphic Encryption (Brakerski et al., 2014; Cheon et al., 2017) allows a computing party to perform an arbitrary number of computational operations over ciphertext data without decrypting it.

The key steps of the process can be summarized as follows: (i) all clients agree on and initialize a FHE keypair, (ii) each client encrypts the local neighbor feature array and sends it to the server, and (iii) upon receiving all encrypted neighbor feature arrays from clients, the server performs secure neighbor feature aggregation

\[ \left[ \sum_{j \in \mathcal{N}_i} A_{ij} x_j \right] = \sum_{k=1}^{K} \left[ \sum_{j \in \mathcal{N}_k} \mathbb{I}_k(e(j)) \cdot A_{ij} x_j \right], \quad (8) \]

where \( [\cdot] \) represents the encryption function. The server then distributes the aggregated neighbor feature array to each client, and (iv) upon receiving the aggregated neighbor feature array, each client decrypts it and moves on to the model training phase.

The aggregation server performs neighbor feature aggregation without having access (i.e., decrypting) to plaintext neighbor features from each client. Each client only receives the accumulation of neighbor features. Differential privacy methods can also be integrated in highly privacy-sensitive applications. Note that with FHE, our secure model gradient aggregation function is also in the encrypted form, which provides extra privacy guarantees. Additionally, we propose an efficient FHE file packing technique, Boolean Packing, which optimizes the communication overhead introduced by cryptographic primitives on a large scale. The encrypted feature aggregation then only requires twice the communication cost of the raw data, which drastically improves the communication overhead compared to general encryption (\( 20 \times \) the communication cost of the raw data).
Packing works by packing arrays of boolean values into integers, and our experimental results on the overhead of FHE can be found in Appendix D.

Algorithm 1 FedGCN Federated Training for Graph Convolutional Network

\[ \text{// Communication Round} \]
for each client \( k \in [K] \) do in parallel
| Send \( \{\sum_{j \in N_k} \mathbb{I}_k(c(j)) \cdot A_{ij} x_j\}_{i \in V_k} \) to the server
end

\[ \text{// Server Operation} \]
for \( i \in V \) do in parallel
| \( \sum_{j \in N_i} A_{ij} x_j = \sum_{k=1}^K [\sum_{j \in N_i} \mathbb{I}_k(c(j)) \cdot A_{ij} x_j] \)
end

for each client \( k \in [K] \) do in parallel
| if 1-hop then
| Receive \( \{\sum_{j \in N_i} A_{ij} x_j\}_{i \in V_k} \) and decrypt it
| end
| if 2-hop then
| Receive \( \{\sum_{j \in N_i} A_{ij} x_j\}_{i \in V_k} \) and decrypt it
| end
end

// Training Round
for \( t = 1, \ldots, T \)
| for each client \( k \in [K] \) do in parallel
| Receive \( \{w^{(t)}\} \) and decrypt it
| Set \( w_k^{(t,1)} = w^{(t)} \)
| for \( e = 1, \ldots, \tau \) do
| Set \( g_{w_k^{(t,e)}} = \nabla w \cdot f_k(w; \theta_k) \)
| \( w_k^{(t,e+1)} = w_k^{(t,e)} - \eta g_{w_k^{(t,e)}} \) // Update Parameters
| end
| \( \Delta_{w_k^{(t,\tau)}} = w_k^{(t,\tau+1)} - w_k^{(t,1)} \)
| Send \( \Delta_{w_k^{(t,\tau)}} \) to the server
end

// Server Operations
\[ \Delta^{(t)} = \frac{1}{K} \sum_{k=1}^K [\Delta_{w_k^{(t,\tau)}}] \] // Difference Aggregation
\[ w^{(t+1)} = w^{(t)} - [\Delta^{(t)}] \] and broadcast to local clients
// Update Global Models

Training Algorithm. Based on the insights in the previous section, we introduce the FedGCN training algorithm shown in Algorithm 1. The algorithm requires communication between clients and the central server at the initial communication round. Clients first send the encrypted accumulations of local node features to the server. The server then accumulates the neighbor features for each node, as described above. Each client receives and decrypts the feature aggregation of its one-hop or two-hop neighbors. After communication, FedGCN uses the standard FedAvg algorithm McMahan et al. (2017) to train the models. Specifically, each client computes \( \tau \) gradient descent steps for local updates. Here we use \( w_k^{(t,1)} \) to denote the concatenation of the weights \( W_k^{(t)} \) across the \( L \) GCN layers, for client \( k \) in global training round \( t \) and local training step \( e \), and \( f_k \) to denote the local loss function, e.g., the cross entropy of the classification estimates. After \( \tau \) local steps, the local model updates at clients are sent to the central server for the global model update, which is again secured through FHE, and the new global model is pushed back to all clients to begin the next training round. The process repeats for \( T \) global rounds until convergence. Note that this procedure can easily be replaced with other federated learning methods, e.g., Reddi et al. (2020); Fallah et al. (2020)’s aggregation methods or local update procedures.

5 FedGCN Convergence and Communication Analysis

In this section, we theoretically analyze the convergence rate and communication cost of FedGCN for i.i.d. and non-i.i.d. data. We also empirically validate our analysis on the SBM (Holland et al., 1983; Abbe, 2017) and real datasets.

5.1 Definition of Data Distribution across Clients

For simplicity, we assume the number of node label classes is equal to the number of clients, \( K \).

Definition 5.1. The data distribution is i.i.d. when nodes are uniformly and randomly assigned to clients with label distribution \( \frac{1}{K}, \ldots, \frac{1}{K} \).

Definition 5.2. The data distribution is non-i.i.d when nodes at a given client have the same class (e.g., \( [1, 0, \ldots, 0]^T \)).

Definition 5.3. The data distribution is partial i.i.d when the node label distribution at a given client is a mixture of i.i.d. and non-i.i.d. data, e.g., \( [1 - p + \frac{p}{K}, \frac{p}{K}, \ldots, \frac{p}{K}]^T \), where \( p \in [0, 1] \) is the i.i.d control parameter.

5.2 Convergence Rate

We first define some notation and assumptions. We use \( ||x|| \) to denote the \( \ell_2 \) norm if \( x \) is a vector, and the Frobenius norm if \( x \) is a matrix. Each client \( k \)’s local loss function is denoted by \( f_k \), while \( f \) denotes the global loss function.

Assumption 5.4. (\( \lambda \)-Lipschitz Continuous Gradient) There exists a constant \( \lambda > 0 \), such that
\[ ||\nabla f_k(w) - \nabla f_k(v)|| \leq \lambda ||w - v||, \forall w, v \in \mathbb{R}^d \text{ and } k \in [K]. \]

Assumption 5.5. (Bounded Global Variability) There exists a constant \( \sigma_G \geq 0 \), such that the global variability of the local gradients of the cost function \( ||\nabla f_k(w_k) - \nabla f(w_k)|| \leq \sigma_G, \forall k \in [K], \forall t. \)
Assumptions 5.4 and 5.5 are standard in the federated learning literature (Yang et al., 2021). We then consider a two-layer GCN, though our analysis can be extended to fit any number of layers. We work from Yang et al. (2021)’s convergence result for federated learning on non-i.i.d. data:

**Theorem 5.6.** (Convergence Rate, Yang et al. (2021)) Let the constant local learning rate \( \eta_L \) and global learning rate \( \eta_G \) be chosen as \( \eta_L \leq \frac{1}{8\alpha} \) and \( \eta_G \eta_L \leq \frac{1}{\lambda^2} \). Under Assumptions 5.4 and 5.5, there exists a constant \( b \) such that \( \min_{i \in [T]} E[\|\nabla f_i(w_i)\|^2] \) can be upper-bounded by

\[
\Phi + \frac{15\tau^2\eta_G^2\lambda^2}{b}\|\nabla f_k(w) - \nabla f(w)\|^2, \tag{9}
\]

where \( \Phi = \frac{f_0 - f_k}{b\eta_L\eta_G\tau^T} \) and \( f_k \) is the local loss function.

The convergence rate is thus bounded by the difference of the gradients of the local and global loss functions \( \|\nabla f_k(w) - \nabla f(w)\| \). We can quantify this difference for 0-, 1-, and 2-hop FedGCN:

**Proposition 5.7.** (Convergence Rates for FedGCN) For generic graph, there exist a constant \( \lambda_c \) such that the difference between the local and global gradients \( \|\nabla f_k(w) - \nabla f(w)\| \) can be upper-bounded by

\[
\lambda_c\|X_k^T A_k^T A_k X_k - Z_{glob}\|, \tag{10}
\]

where

\[
Z_{glob} = X^T A^T A X A X A X A X \ldots A X A X A X A X . \tag{11}
\]

Table 1 bounds \( \|\nabla f_k(w) - \nabla f(w)\| \) for the FedGCN models for generic graphs. In the table \( A_k \) and \( \bar{A}_k \) respectively denote the adjacency matrix at client \( k \) with 1- and 2-hop communication, similarly for \( \bar{X}_k \) and \( \bar{X}_k \). In the table, we give the approximate values of these bounds for a SBM model when \( \alpha, \mu \ll 1 \).

Appendix B proves this result. We can then analyze the convergence of FedGCN with different levels of communication, simply by knowing the node features and graph adjacency matrix (i.e., without knowing the model). Intuitively, more communication makes the difference between local and global gradients smaller. This effect, however, is difficult to quantify for an arbitrary graph topology. Thus, the table further gives approximate expressions for the SBM model, in which we assume \( N \) nodes with \( d \)-dimensional node features and \( K \) classes, and \( N \) nodes are partitioned across \( K \) clients. Nodes in the same (different) class have an edge between them with probability \( \alpha \) \( (\mu \alpha) \), \( \mu \in [0, 1] \).

Appendix A details the full SBM model.

Examining Table 1’s SBM bounds, we observe that for a fixed number of hops, the convergence is faster as the data become more i.i.d., by a factor of \( \alpha \). Similarly, the convergence becomes faster with more communication hops, by a factor that depends on \( K, N, \alpha \), and \( \mu \). As \( K \) grows, for example, the differences in different hops of communication become more pronounced by a factor of \( O(K^{-\alpha}) \), which we would intuitively expect as a larger \( K \) leads to more cross-client edges. We validate these results in Section 6.3.

### 5.3 Communication Cost and Tradeoffs

We next examine the communication cost, and the convergence-communication tradeoff, of FedGCN. As for the convergence analysis, we derive communication costs for general graphs and then more interpretable results for the SBM model.

**Proposition 5.8.** (Communication Cost for FedGCN) For L-layer GCNs, the size of messages from clients to the server in a generic graph is

\[
\sum_{i \in V} |c(N_i)|d + \sum_{k=1}^{K} |N_k^{i-1}|d, \tag{12}
\]

where \( c(N_i) \) denotes the set of clients storing the neighbors of node \( i \). For data distribution with a factor \( p \) on SBM model, the expected size of the first part, \( \sum_{i \in V} |c(N_i)|d \), is

\[
N(1 + (K - 1)(1 - \alpha)) \frac{\alpha} {2^T} (1 - \mu \alpha)^{\frac{N(K-1)}{2^T}} d. \tag{13}
\]

For better understanding of the above form, Table 2 gives the approximated (assuming \( \alpha, \mu \ll 1 \)) 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 C 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, \( c_p N d \) 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 1’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 the next section.

### 6 Experimental Validation

We validate FedGCN’s performance relative to previously proposed algorithms on multiple real datasets. FedGCN converges faster, to a more accurate model and with less
FedGCN: Convergence and Communication Tradeoffs in Federated Training of Graph Convolutional Networks

| Data Distribution | 0-hop | 1-hop | 2-hop |
|-------------------|-------|-------|-------|
| Generic Graph     | \(X_k A_k^1 A_k X_k - Z_{glob}\) | \(X_k A_k^1 A_k A_k X_k - Z_{glob}\) | \(X_k A_k^1 A_k A_k A_k X_k - Z_{glob}\) |
| Non-i.i.d. (SBM)  | \(1 - \frac{1}{K}\) \(\sum_{k=1}^{K} \|B_k\|^2 + \sigma\) | \(1 - \frac{1}{K}\) \(\sum_{k=1}^{K} \|B_k\|^2 + \|B_k\|^2 + \sigma\) | \(1 - \frac{1}{K}\) \(\sum_{k=1}^{K} \|B_k\|^2 + \|B_k\|^2 + \sigma\) |
| Partial-i.i.d. (SBM) | \(1 - \frac{1}{K}\) \(\sum_{k=1}^{K} \|B_k\|^2 + \sigma\) | \(1 - \frac{1}{K}\) \(\sum_{k=1}^{K} \|B_k\|^2 + \|B_k\|^2 + \sigma\) | \(1 - \frac{1}{K}\) \(\sum_{k=1}^{K} \|B_k\|^2 + \|B_k\|^2 + \sigma\) |
| i.i.d. (SBM)      | \(1 - \frac{1}{K}\) \(\sum_{k=1}^{K} \|B_k\|^2\) | \(1 - \frac{1}{K}\) \(\sum_{k=1}^{K} \|B_k\|^2 + \|B_k\|^2\) | \(1 - \frac{1}{K}\) \(\sum_{k=1}^{K} \|B_k\|^2 + \|B_k\|^2\) |

Table 1. Convergence rate bounds of FedGCN under different communication levels. A (double) dot above a variable denotes its value with (2-) 1-hop communication. We define \(c_o = \frac{1-\mu N}{N(K-1)}\), \(c_p = \frac{1+\mu N}{N(K-1)}\) for the SBM model. The bound is smaller (convergence is faster) with more hops of communication. When the i.i.d control parameter \(p\) increases, the difference between the local graph and the 2-hop graph increases, which means communication helps more when data is more i.i.d. \(\sigma\) measures the difference of the local distribution and global distribution. Non-i.i.d. data implies a longer convergence time.

| Data Distribution | 0-hop | 1-hop | 2-hop |
|-------------------|-------|-------|-------|
| Generic Graph     | 0     | \(\sum_{i \in V} |c(N_i)|d + Nd\) | \(\sum_{i \in V} |c(N_i)|d + \sum_{k=1}^{K} |N_k|d\) |
| Non-i.i.d. (SBM)  | 0     | \((c_o + 2)Nd\) | \(2(c_o + c_p + 1)Nd\) |
| Partial-i.i.d. (SBM) | 0     | \((c_o p + c_o + 2)Nd\) | \(2(c_o p + c_o + 1)Nd\) |
| i.i.d. (SBM)      | 0     | \((c_o + c_o + 2)Nd\) | \(2(c_o + c_o + 1)Nd\) |

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

We compare the performance of five training methods: **Centralized GCN** assumes a single client has access to the entire graph. It has neither information loss nor communication; **Distributed GCN** (Scardapane et al., 2020) trains GCN in distributed clients which requires communicating node features and hidden states of each layer. **FedGCN (0-hop)** (Section 4) is equivalent to federated training without communication (FedGraphnn) (Wang et al., 2020a; Zheng et al., 2021; He et al., 2021a). **BDS-GCN** (Wan et al., 2022) randomly samples cross-client edges in each global training round, while **FedSage+** (Zhang et al., 2021) recovers missing neighbors by learning a linear predictor based on the node embedding, using cross-client information in each training round. It is thus an approximation of FedGCN (1-hop), which communicates the 1-hop neighbors’ information across clients to reduce information loss with less communication. FedGCN eliminates information loss for two-layer GCNs by communicating the two-hop neighbors’ information across clients.

We consider an i.i.d. data distribution, in which nodes are partitioned across clients uniformly at random, and a non-i.i.d. distribution, in which each client only contains nodes with the same (randomly chosen) label. Partial-i.i.d. settings sample a fraction of data in each manner. We use a two-layer GCN with ReLU activation for the first and Softmax for the second layer, as in Kipf & 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 \(5 \times 10^4\) and 3 local steps per round for federated settings. For the OGBN-Arxiv dataset, we instead use 256 hidden units and 500 training rounds. The adjacency matrix is normalized by row degree \(\hat{A} = D^{-1}A\) to provide equal gradient updates. We set the number of clients to equal the number of classes and average over 10 experiment runs. Extended results, including an evaluation of the overhead of FHE, are in the appendix.

6.2 Effect of Cross-Client Communication

We first evaluate our methods under i.i.d., non-i.i.d., and partial (50%) i.i.d. data distributions on the Cora dataset to illustrate FedGCN’s performance relative to the centralized and BDS-GCN baselines under different levels of communication. As shown in Figure 3, FedGCN converges much faster and to a higher test and training accuracy compared...
to BDS-GCN in all settings. We omit the convergence of the centralized and distributed GCN training for ease of visualization in the figure, as they can be expected to converge faster than all federated GCN algorithms, since they use full node information in each iteration. The appendix (for Cora, Citeseer, and Pubmed) and Figure 4(for OGBN-ArXiv) show that FedGCN’s variants consistently achieve higher accuracy that that of centralized or distributed GCN, with the optimal number of hops depending on the data distribution. Similarly, since FedSage+ approximates FedGCN (1-hop)’s model, we do not show its accuracy results separately in Figure 3. The FedGCN (0-hop) performs worst in the i.i.d. and partial i.i.d. settings, due to information loss from cross-client edges. Under the extreme non-i.i.d. setting, the information loss is small, as there are few cross-client edges. The performance of all algorithms is thus similar, indicating that FedGCN (0-hop) has sufficient information to train a good model.

Convergence time. We define the convergence time as the index of the first global training round when the validation accuracy does not change more than 0.01 compared with that of the last round. Table 4 shows the convergence time for Cora, Citeseer, and Pubmed. FedGCN has the lowest convergence time, about 50% lower than BDS-GCN, for all three datasets in the i.i.d. and partial-i.i.d. settings. In the non-i.i.d. setting, FedGCN converges more slowly due to including information on cross-client nodes from other classes (which is less helpful for the model). FedGCN (0-hop) converges slowly due to information loss, as does BDS-GCN. The centralized algorithm, although it does not have any information loss, converges more slowly than FedGCN or FedGCN (1-hop) as all federated methods contain \( \tau = 3 \) local training epochs in each global training round and can thus make more progress in each round.

Communication cost with accuracy. Figure 4 shows the communication cost and test accuracy of different methods on the OGBN-ArXiv dataset. FedGCN (0-, 1-, and 2-hop) requires little communication with high accuracy, while Distributed GCN, BDS-GCN and FedSage+ require communication at every round, incurring over 100× the communication cost of any of FedGCN’s variants. FedGCN (0-hop) requires much less communication than 1- and 2-hop FedGCN, but has lower accuracy due to information loss in the i.i.d. and partial-i.i.d. settings, displaying a convergence-communication tradeoff. Both 1- and 2-hop FedGCN achieve similar accuracy as centralized GCN, indi-

Table 4. Convergence time on three datasets, for i.i.d., non-i.i.d., and 50% i.i.d. data. FedGCN performs best on i.i.d. and partially i.i.d. data, where FedGCN (0-hop) has the most information loss. Compare with BDS-GCN, our algorithm converges 51.7% faster.

| Method         | Cora       | Citeseer   | Pubmed     |
|----------------|------------|------------|------------|
|                | i.i.d.     | partial i.i.d. | non-i.i.d. | i.i.d.     | partial i.i.d. | non-i.i.d. | i.i.d.     | partial i.i.d. | non-i.i.d. |
| Central, DGCN  | 97         | 97         | 97         | 121        | 121         | 121         | 58         | 58         | 58         |
| FedGCN (0-hop) | 235        | 232        | 87         | 157        | 121         | 165         | 184        | 92         | 54         |
| BDS-GCN        | 128        | 94         | 64         | 105        | 110         | 150         | 79         | 75         | 130        |
| FedGCN (1-hop) | 40         | 60         | 62         | 63         | 92          | 111         | 77         | 57         | 31         |
| FedGCN         | 32         | 42         | 77         | 39         | 51          | 87          | 35         | 50         | 30         |
FedGCN: Convergence and Communication Tradeoffs in Federated Training of Graph Convolutional Networks

Figure 4. Test accuracy vs. communication cost until convergence of different algorithms in the i.i.d., partial-i.i.d. and non-i.i.d. settings for the OGBN-ArXiv dataset. FedGCN uses orders of magnitude less communication (at least 100×) than BDS-GCN and FedSage+, while achieving higher test accuracy.

Figure 5. Convergence time (left), communication cost (middle) on Cora, and theoretical convergence upper bound (right, Table 1). FedGCN (1-hop) balances convergence and communication.

6.3 Validation of Theoretical Analysis

We validate the qualitative results in Propositions 5.7 and 5.8 on the Cora dataset. As shown in Figure 5, 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 5 shows Table 1’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 5.6 the convergence time increases with \(|\nabla f_k(w_k) - \nabla f(w)|\). 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 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.

7 CONCLUSION

We propose FedGCN, a framework for federated training of graph convolutional networks for semi-supervised node classification. The FedGCN training algorithm is based on the insight that, although distributed GCN training typically ignores cross-client edges, these edges can in practice contain information useful to the model. Moreover, evaluating and training the GCN model requires only a single round of communication before training begins. FedGCN allows for different levels of communication to accommodate different privacy and overhead concerns, with more communication generally leading to less information loss and faster convergence, and further integrates FHE for additional privacy protection. We quantify FedGCN’s convergence under different levels of communication and different degrees of non-i.i.d. data across clients and show that FedGCN achieves high accuracy on real datasets, with orders of magnitude less communication than previous algorithms. One-hop FedGCN balances convergence and communication costs.

Although FedGCN reduces the communication cost compared to prior federated graph training algorithms without information loss, our analytical framework suggests that new algorithms, e.g., selectively communicating the edge information that is most likely to accelerate convergence, could better optimize the tradeoff between fast convergence
and little communication. Optimizing other hyperparameters like the number of local training epochs, which affects the convergence of FedGCN in non-i.i.d. settings, can further accelerate convergence but come with their own communication costs that may be quantified and optimized by building on our analytical framework.

**REFERENCES**

Abbe, E. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research*, 18(1):6446–6531, 2017.

Benamira, A., Devillers, B., Lesot, E., Ray, A. K., Saadi, M., and Malliaros, F. D. Semi-supervised learning and graph neural networks for fake news detection. In 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 568–569. IEEE, 2019.

Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Cheon, J. H., Kim, A., Kim, M., and Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In International conference on the theory and application of cryptology and information security, pp. 409–437. Springer, 2017.

Fallah, A., Mokhtari, A., and Ozdaglar, A. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems*, 33:3557–3568, 2020.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 1025–1035, 2017.

He, C., Balasubramanian, K., Ceyani, E., Yang, C., Xie, H., Sun, L., He, L., Yang, L., Yu, P. S., Rong, Y., et al. Fed-graphnn: A federated learning system and benchmark for graph neural networks. *arXiv preprint arXiv:2104.07145*, 2021a.

He, C., Ceyani, E., Balasubramanian, K., Annavaram, M., and Avestimehr, S. Spreadgnn: Serverless multi-task federated learning for graph neural networks. *arXiv preprint arXiv:2106.02743*, 2021b.

Holland, P. W., Laskey, K. B., and Leinhardt, S. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.
Keriven, N. and Vaiter, S. Sparse and smooth: improved guarantees for spectral clustering in the dynamic stochastic block model. arXiv preprint arXiv:2002.02892, 2020.

Keriven, N., Bietti, A., and Vaiter, S. Convergence and stability of graph convolutional networks on large random graphs. arXiv preprint arXiv:2006.01868, 2020.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

Lei, J. and Rinaldo, A. Consistency of spectral clustering in stochastic block models. The Annals of Statistics, 43(1): 215–237, 2015.

Li, G., Muller, M., Thabet, A., and Ghanem, B. Deepgcns: Can gcns go as deep as cnns? In Proceedings of the IEEE/CVF international conference on computer vision, pp. 9267–9276, 2019a.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. arXiv preprint arXiv:1907.02189, 2019b.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics, pp. 1273–1282. PMLR, 2017.

Ni, X., Xu, X., Lyu, L., Meng, C., and Wang, W. A vertical federated learning framework for graph convolutional network. arXiv preprint arXiv:2106.11593, 2021.

PALISADE. Palisade release, 2020. URL https://gitlab.com/palisade/palisade-release.

Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. arXiv preprint arXiv:2003.00295, 2020.

Scardapane, S., Spinelli, I., and Di Lorenzo, P. Distributed graph convolutional networks. arXiv preprint arXiv:2007.06281, 2020.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. AI magazine, 29(3):93–93, 2008.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. arXiv preprint arXiv:1710.10903, 2017.

Wan, C., Li, Y., Li, A., Kim, N. S., and Lin, Y. Bns-gcn: Efficient full-graph training of graph convolutional networks with partition-parallelism and random boundary node sampling. Proceedings of Machine Learning and Systems, 4:673–693, 2022.

Wang, B., Li, A., Li, H., and Chen, Y. Graphfl: A federated learning framework for semi-supervised node classification on graphs. arXiv preprint arXiv:2012.04187, 2020a.

Wang, K., Shen, Z., Huang, C., Wu, C.-H., Dong, Y., and Kanakia, A. Microsoft academic graph: When experts are not enough. Quantitative Science Studies, 1(1):396–413, 2020b.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q., and Poor, H. V. Federated learning with differential privacy: Algorithms and performance analysis. IEEE Transactions on Information Forensics and Security, 15:3454–3469, 2020.

Yang, H., Fang, M., and Liu, J. Achieving linear speedup with partial worker participation in non-iid federated learning. arXiv preprint arXiv:2101.111203, 2021.

Zhang, K., Yang, C., Li, X., Sun, L., and Yiu, S. M. Subgraph federated learning with missing neighbor generation. Advances in Neural Information Processing Systems, 34, 2021.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. An end-to-end deep learning architecture for graph classification. In Thirty-second AAAI conference on artificial intelligence, 2018.

Zhang, W., Liu, H., Liu, Y., Zhou, J., and Xiong, H. Semi-supervised hierarchical recurrent graph neural network for city-wide parking availability prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pp. 1186–1193, 2020.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. Federated learning with non-iid data. arXiv preprint arXiv:1806.00582, 2018.

Zheng, L., Zhou, J., Chen, C., Wu, B., Wang, L., and Zhang, B. Asfgnn: Automated separated-federated graph neural network. Peer-to-Peer Networking and Applications, 14 (3):1692–1704, 2021.

Zhou, Z. and Amini, A. A. Analysis of spectral clustering algorithms for community detection: the general bipartite setting. The Journal of Machine Learning Research, 20 (1):1774–1820, 2019.
A.1 Accuracy with Different Client Numbers

Figure 6 shows the performance of methods with changing client numbers. FedGCN(2-hop) has stable performance with the change of the client numbers since it fully recovers the cross-client edges by communication and does not have information loss. The accuracy of FedGCN(0-hop) decreases with the number of clients since it drops more cross-client edges with the increase of client number. The drop becomes more serious in the i.i.d. condition.

A.2 Performance of GCN with Many Layers

Many GCNs used in practice for node classification only contain two layers. As shown in Figure 7 from (Kipf & Welling, 2016), 2 or 3 layers is enough to have the best performance on some datasets. For models deeper than 7 layers, training without the use of residual connections can become difficult, as the effective context size for each node increases by the size of its $L$-hop neighbors (for a model with $L$ layers) with each additional layer. We also reproduce these results on Cora dataset. As shown in Figure 8, the accuracy of the GCN model does not increase with the number of layers beyond 2 or 3 layers, which exhibit the highest accuracies.

A.3 Performance of FedGCN with Many Layers

As shown in Figure 9, the 2-hop FedGCN is stable even for 5-layer GCNs. It then becomes hard to train the model as the effective context size for each node increases by the size of its $L$-hop neighbors (for a model with $L$ layers) with each additional layer, which is consistent with the centralized model in Figure 8.

If $L$ is too large, then the $L$-hop neighbors may actually consist of almost the entire graph (many social network graphs have diameters $<10$), which might introduce prohibitive storage and communication requirements when there are many clients.
Figure 7. Performance of GCNs with different layers on Citeseer, Cora and Pubmed Dataset. The figure is from (Kipf & Welling, 2016).

Figure 8. Performance of GCNs with different layers on Cora Dataset during train, validation, and test process.

A.4 Federated Learning

Federated learning is first proposed in (McMahan et al., 2017), which builds 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 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 $w_t$ is sent to $K$ client devices.

Every client $k$ computes the average gradient, $g_k$, on its local data by using the current model $w_k^i$ with $E$ local epochs. For a client learning rate $\eta$, the local client update of 1 local epoch, $w_k$, is given by

$$w_k \leftarrow w_k - \eta g_k.$$  \hfill (14)

The server then does a weighted aggregation of the client local models to obtain a new global model,

$$w_{t+1} = \frac{1}{\sum_{k=1}^{K} n_k} \sum_{k=1}^{K} n_k w_k^t,$$  \hfill (15)

where $n_k$ is the number of local data points in client $k$. 
A.5 Graph Convolutional Network

A multi-layer Graph Convolutional Network (GCN) (Kipf & Welling, 2016) with row normalization has the layer-wise propagation rule

\[ H^{(l+1)} = \phi(D^{-1/2}AH^{(l)}W^{(l)}) \]

where \( \tilde{A} = A + I_N \), \( I_N \) is the identity matrix, \( D_{ii} = \sum_j \tilde{A}_{ij} \) and \( W^{(l)} \) is a layer-specific trainable weight matrix. The activation function is \( \phi \), 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 \( H^{(l)} \in \mathbb{R}^{N \times D} \), which contains high-level representations of the graph nodes transformed from the initial features; \( H^{(0)} = X \).

In general, for a GCN with \( L \) layers of the form 16, the output for node \( i \) will depend on neighbours up to \( L \) steps away. We denote this set by \( N_i^L \) as \( L \)-hop neighbors of \( i \). Based on this idea, the clients can first communicate the information of nodes. After communication of information, we can then train the model.

A.6 Stochastic Block Model

For positive integers \( K \) and \( n \), a probability vector \( p \in [0, 1]^K \), and a symmetric connectivity matrix \( B \in [0, 1]^{K \times K} \), the SBM defines a random graph with \( n \) nodes split into \( K \) 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 \( \{1, \ldots, K\} \) according to the distribution \( p \); 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 \( i \) and \( j \) with probability \( B_{ij} \), where the \((i, j)\) entry of \( B \) is

\[ B_{ij} = \begin{cases} 
\alpha, & i = j \\
\mu \alpha, & i \neq j 
\end{cases} \]

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 \( \alpha \) (which is the same for each class) and the edge formation probability between nodes in different classes is \( \mu \alpha \).

Let \( Y \in \{0, 1\}^{n \times K} \) denote the matrix representing the nodes’ class memberships, where \( Y_{ik} = 1 \) indicates that node \( i \) belongs to the \( k \)-th class, and is 0 otherwise. We use \( A \in \{0, 1\}^{n \times n} \) to denote the (symmetric) adjacency matrix of the graph, where \( A_{ij} \) indicates whether there is a connection (edge) between node \( i \) and node \( j \). From our node connectivity model, we find that given \( Y \), for \( i < j \), we have

\[ A_{ij} | (Y_{ik} = 1, Y_{jk} = 1) \sim \text{Ber}(B_{kl}), \]

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

A.7 Computation Resource

All experiments are done in a single machine. Normal computers can run the experiments and reproduce the results within a few hours.
B Convergence Proof

We first analyze a 1-layer GCN, then we mainly analysis a 2-layer GCN, which is the most common architecture for graph neural network. Our analysis also fits any layers of GCN and GraphSage.

We then assume the number of classes equals to the number of clients, \( M = K \), for simplicity of analysis.

B.1 Convergence Analysis of 1-layer GCNs

B.1.1 Gradient of Centralized GCN

For graph \( G \) with adjacency matrix \( A \) and feature matrix \( X \), we consider a 1-layer graph convolutional network with Softmax activation and cross entropy loss, which has the following form

\[
Z = AXW, \tag{19}
\]

\[
Q = \phi(Z), \tag{20}
\]

where

\[
Q_{ij} = \frac{e^{Z_{ij}}}{\sum_{k=1}^{K} e^{Z_{ik}}}, \tag{21}
\]

Let \( f \) represent the loss function

\[
f = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} Y_{ij} \log Q_{ij}. \tag{22}
\]

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

Proof. At first, we calculate the gradient of \( f \) given the element \( Z_{ij} \) of the matrix \( Z \), \( \frac{\partial f}{\partial Z_{ij}} \),

\[
\frac{\partial f}{\partial Z_{ij}} = \frac{\partial(-\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} Y_{ij} \log Q_{ij})}{\partial Z_{ij}}
\]

\[
= \frac{\partial(-\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} Y_{ij} \log \frac{e^{Z_{ij}}}{\sum_{k=1}^{K} e^{Z_{ik}}})}{\partial Z_{ij}}
\]

\[
= \frac{\partial(-\frac{1}{N} \sum_{j=1}^{K} Y_{ij} \log \frac{e^{Z_{ij}}}{\sum_{k=1}^{K} e^{Z_{ik}}})}{\partial Z_{ij}}
\]

\[
= -\frac{1}{N} \sum_{j=1}^{K} \frac{\partial Y_{ij} \log \frac{e^{Z_{ij}}}{\sum_{k=1}^{K} e^{Z_{ik}}}}{\partial Z_{ij}}
\]

\[
= \frac{1}{N} \frac{\partial}{\partial Z_{ij}} \left( Y_{ij} Z_{ij} - Y_{ij} \log \sum_{k=1}^{K} e^{Z_{ik}} \right)
\]

\[
= \frac{1}{N} (Y_{ij} - \frac{\partial}{\partial Z_{ij}} \left( Y_{ij} \log \sum_{k=1}^{K} e^{Z_{ik}} \right))
\]

\[
= \frac{1}{N} (Y_{ij} \left( 1 - \frac{\partial}{\partial Z_{ij}} \left( \log \sum_{k=1}^{K} e^{Z_{ik}} \right) \right))
\]

\[
= \frac{1}{N} (Y_{ij} \left( 1 - \frac{e^{Z_{ij}}}{\sum_{k=1}^{K} e^{Z_{ik}}} \right))
\]

\[
= \frac{1}{N} (Y_{ij} \left( \frac{e^{Z_{ij}}}{\sum_{k=1}^{K} e^{Z_{ik}}} - Y_{ij} \right))
\]

\[
= \frac{1}{N} (Q_{ij} - Y_{ij})
\]
Given the property of matrix, we have
\[ \frac{\partial f}{\partial Z} = \frac{1}{N}(Q - Y). \]

**Lemma 1** If \( Z = AXB \),
\[ \frac{\partial f}{\partial X} = A^T \frac{\partial f}{\partial Z} B^T. \]

**Equation 2** The gradient over the weights of GCN
\[ \frac{\partial f}{\partial W} = \frac{1}{N} X^T A^T (\phi(AXW) - Y). \] (24)

**Proof.**
\[ \frac{\partial f}{\partial W} = (AX)^T \frac{\partial f}{\partial Z} \]
\[ = X^T A^T \frac{\partial f}{\partial Z} \]
\[ = \frac{1}{N} X^T A^T (Q - Y) \]
\[ = \frac{1}{N} X^T A^T (\phi(AXW) - Y) \] (25)

**B.1.2 Gradients of local models**
We then consider the federated setting. Let \( A^{N \times N} \) denotes the adjacency matrix of all nodes and \( A_k^{N_k \times N_k} \) denotes the adjacency matrix of the nodes in client \( k \). Let \( f_k \) represents the local loss function (without communication) of client \( k \). Then the local gradient is
\[ \frac{\partial f_k}{\partial W} = \frac{1}{N_k} X_k^T A_k^T (\phi(A_k X_k W) - Y_k) \] (26)

**B.1.3 Gradients of local models with 1-hop communication**
With 1-hop communication, let \( \hat{A}_k^{N_k \times |N_k|} \) denotes the adjacency matrix of the nodes in client \( k \) and their 1-hop neighbors (\( N_k \) also includes the current nodes). The output of GCN with 1-hop communication (recovering 1-hop neighbor information) is
\[ \phi(\hat{A}_k \hat{X}_k W). \] (27)

**B.1.4 Bound the difference of local gradient and global gradient**
Assuming each client has equal number of nodes, we have \( N_k = \frac{N}{K} \). By assuming the linearity of the activation function, we can then provide the following approximation to compare the local gradient and the global gradient
\[ \| \frac{\partial f_k}{\partial W} - \frac{\partial f}{\partial W} \| \leq \lambda \| K X_k^T A_k^T A_k X_k - X^T A^T AX \|, \] (28)
where \( \lambda \) is a constant.

Let \( \hat{f}_k \) represents the loss function with 1-hop communication, the difference between local gradient with 1-hop communication and the global gradient is
\[ \| \frac{\partial \hat{f}_k}{\partial W} - \frac{\partial f}{\partial W} \| \leq \lambda \| K \hat{X}_k^T \hat{A}_k \hat{X}_k - X^T A^T AX \|. \] (29)
B.2 Convergence Analysis of 2-layer GCNs

B.2.1 Gradient of Centralized GCN

Based on the analysis of 1-layer GCNs, for graph $G$ with adjacency matrix $A$ and feature matrix $X$ 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

$$Z = A\phi_1(AXW_1)W_2,$$  \hspace{1cm} (30)

$$Q = \phi_2(Z),$$  \hspace{1cm} (31)

where

$$Q_{ij} = \frac{e^{Z_{ij}}}{\sum_{k=1}^{K} e^{Z_{ik}}}$$  \hspace{1cm} (32)

$$f = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} Y_{ij} \log Q_{ij}.$$  \hspace{1cm} (33)

We then show how to calculate the gradient $\nabla f(w) = [\frac{\partial f}{\partial W_1}, \frac{\partial f}{\partial W_2}]$.

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

**Equation 2** The gradient over the weights of the second layer

$$\frac{\partial f}{\partial W_2} = \frac{1}{N} (\phi_1(W_1^T X^T A^T)) A^T (\phi_2(A\phi_1(AXW_1)W_2) - Y)$$  \hspace{1cm} (34)

*Proof.*

$$\frac{\partial f}{\partial W_2} = (A\phi_1(AXW_1))^T \frac{\partial f}{\partial Z}$$

$$= (\phi_1(AXW_1))^T A^T \frac{\partial f}{\partial Z}$$

$$= \frac{1}{N} (\phi_1(AXW_1))^T A^T (Q - Y)$$

$$= \frac{1}{N} (\phi_1(AXW_1))^T A^T (\phi_2(A\phi_1(AXW_1)W_2) - Y)$$

$$= \frac{1}{N} (\phi_1(W_1^T X^T A^T)) A^T (\phi_2(A\phi_1(AXW_1)W_2) - Y)$$  \hspace{1cm} (35)

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

$$\frac{\partial f}{\partial W_1} = \frac{1}{N} (A\phi'_1(AXW_1)AX)^T (\phi_2(A\phi_1(AXW_1)W_2) - Y)W_2^T$$  \hspace{1cm} (36)

*Proof.*

$$\frac{\partial f}{\partial W_1} = (A\phi'_1(AXW_1)AX)^T \frac{\partial f}{\partial Z} W_2^T$$

$$= (A\phi'_1(AXW_1)AX)^T \frac{\partial f}{\partial Z} W_2^T$$

$$= \frac{1}{N} (A\phi'_1(AXW_1)AX)^T (Q - Y)W_2^T$$

$$= \frac{1}{N} (A\phi'_1(AXW_1)AX)^T (\phi_2(A\phi_1(AXW_1)W_2) - Y)W_2^T$$  \hspace{1cm} (37)
FedGCN: Convergence and Communication Tradeoffs in Federated Training of Graph Convolutional Networks

B.2.2 Gradient of FedGCN

For client \( k \) with local adjacency matrix \( A_k \), let \( \hat{A}_k^{N_k \times |N_k|} \) denotes the adjacency matrix of the current nodes with complete edge information form their 1-hop neighbors (\( N_k \) also includes the current nodes), and \( \hat{A}_k^{N_k \times |N_k^2|} \) denotes the adjacency matrix of nodes with complete edge information form their 2-hop neighbors (\( N_k^2 \) also includes the current nodes and 1-hop neighbors).

The output of GCN without communication is

\[
\phi_2(A_k \phi_1(A_k X_k W_1) W_2).
\]

(38)

The output of GCN with 1-hop communication is

\[
\phi_2(A_k \phi_1(\hat{A}_k \hat{X}_k W_1) W_2).
\]

(39)

The output of GCN with 2-hop communication is

\[
\phi_2(\hat{A}_k \phi_1(\hat{A}_k \hat{X}_k W_1) W_2).
\]

(40)

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.

\[
\frac{\partial \hat{f}_k}{\partial W} = \frac{1}{N_k} (\hat{A}_k \phi'_1(\hat{A}_k \hat{X}_k W_1) \hat{A}_k \hat{X}_k) (\phi_2(\hat{A}_k \phi_1(\hat{A}_k \hat{X}_k W_1) W_2) - Y_k) W_2^T.
\]

(41)

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

Assuming each client has equal number of nodes, we have \( N_k = \frac{N}{K} \). By assuming the linearity of the activation function, we can then provide the following approximations between the local model and global model.

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

\[
\| \frac{\partial f_k}{\partial W} - \frac{\partial f}{\partial W} \| \leq \lambda \| K X_k^T A_k^T A_k X_k - X^T A^T A X \|.
\]

(42)

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

\[
\| \frac{\partial \hat{f}_k}{\partial W} - \frac{\partial f}{\partial W} \| \leq \lambda \| K \hat{X}_k^T \hat{A}_k^T \hat{A}_k \hat{X}_k - X^T A^T A X \|
\]

(43)

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

\[
\| \frac{\partial \hat{f}_k}{\partial W} - \frac{\partial f}{\partial W} \| \leq \lambda \| K \hat{X}_k^T \hat{A}_k^T \hat{A}_k \hat{X}_k - X^T A^T A X \|
\]

(44)

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

B.3 Analysis on Stochastic Block Model with Node Features

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

Assume the node feature vector \( x \) follows the Gaussian distribution with linear projection \( H \) of node label \( y \),

\[
x \sim \mathcal{N}(Hy, \sigma^2),
\]

(45)

we then have the expectation of the feature matrix

\[
E(X) = E(YH^T).
\]

(46)
According to the Stochastic Block Model, we have
\[ E(A) = P = YBY^T \] (47)

The expectation of the former gradient given the label matrix \( Y \) is then
\[ E(X^TAAX|Y) = H^TYBY^TYBY^TYBY^TYH^T \] (48)

Notice that \( Y^TY \) is counting the number of nodes belonging to each class. Based on this observation, we can better analyze the data distribution.

**B.3.1 Quantify the gradient difference**

For adjacency matrix without communication
\[ E(A_k) = Y_kBY_k^T \] (49)

The expectation of the former gradient given the label matrix \( Y \) is then
\[ E(X_k^TAA_kX_k|Y) = H_k^TY_k^TY_kBY_k^TY_kBY_k^TY_kBY_k^TY_kBY_k^TY_kBY_k^TY_kH_k^T \] (50)

For adjacency matrix with 1-hop communication
\[ E(\hat{A}_k) = Y_kBY_k^T \] (51)

The expectation of the former gradient with 1-hop communication given the label matrix \( Y \) is then
\[ E(\hat{X}_k^T\hat{A}_k\hat{A}_k\hat{X}_k|Y) = H_k^T\hat{Y}_k^T\hat{Y}_kBY_k^T\hat{Y}_kBY_k^T\hat{Y}_kBY_k^T\hat{Y}_kBY_k^T\hat{Y}_kH_k^T \] (52)

For adjacency matrix with 2-hop communication
\[ E(\tilde{A}_k) = \tilde{Y}BY_k^T. \] (53)

The expectation of the former gradient with 1-hop communication given the label matrix \( Y \) is then
\[ E(\tilde{X}_k^T\tilde{A}_k\tilde{A}_k\tilde{X}_k|Y) = H_k^T\tilde{Y}_k^T\tilde{Y}_kBY_k^T\tilde{Y}_kBY_k^T\tilde{Y}_kBY_k^T\tilde{Y}_kBY_k^T\tilde{Y}_kH \] (54)

The difference of gradient can then be written as
\[ \| \frac{\partial f_k}{\partial W} - \frac{\partial f}{\partial W} \| \leq \lambda \| K\hat{Y}_k^T\hat{Y}_kBY_k^T\hat{Y}_kBY_k^T\hat{Y}_kBY_k^T\hat{Y}_kBY_k^T\hat{Y}_k - YTYBY^TYBY^TYBY^TY \| \] (55)

Notice that \( Y_k^TY_k \) is counting the number of nodes in client \( k \) belonging to each class, \( \hat{Y}_k^T\hat{Y}_k \) is counting the number of 1-hop neighbors of nodes in client \( k \) belonging to each class, and \( \tilde{Y}_k^T\tilde{Y}_k \) is counting the number of 2-hop neighbors of nodes in client \( k \) belonging to each class. We then start analyzing the data distribution with labels.

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

**B.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 \( i \)
\[ 1 - (1 - \alpha)^\frac{n}{\alpha^2} (1 - \mu\alpha)^\frac{(n-1)\alpha}{\alpha^2} \] (56)
FedGCN: Convergence and Communication Tradeoffs in Federated Training of Graph Convolutional Networks

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

\[
\frac{N}{K} + \frac{K-1}{K} N \left(1 - (1 - \alpha)^{\frac{N}{K}} (1 - \mu \alpha)^{\frac{N}{K}}\right) \approx \frac{N}{K} + \frac{K-1}{K} N \left(1 - (1 - \alpha) N \frac{N}{K^2} (1 - \mu \alpha (K-1) N)\right)
\]

\[
\approx \frac{N}{K} + \frac{K-1}{K} N \left(1 - (1 - \alpha) N \frac{N}{K^2} - \mu \alpha (K-1) N\right)
\]

\[
= \frac{N}{K} + \frac{K-1}{K} N \left(\alpha N \frac{N}{K^2} + \mu \alpha (K-1) N\right)
\]

\[
= \frac{N}{K} \left(1 + (K-1) (\alpha N \frac{N}{K^2} + \mu \alpha (K-1) N)\right)
\]

(57)

Notice that it is \((1 + (K-1) (\alpha N \frac{N}{K^2} + \mu \alpha (K-1) N))\) 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.

\[
\frac{N}{K} (1 + (K-1) (\alpha N \frac{N}{K^2} + \mu \alpha (K-1) N))
\]

(58)

B.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)

\[
\frac{N}{K} + \frac{K-1}{K} N \left(1 - (1 - \mu \alpha)^{\frac{N}{K}}\right) \approx \frac{N}{K} + \frac{K-1}{K} N \mu \alpha \frac{N}{K}
\]

(59)

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

\[
\frac{N}{K} \left(1 + (K-1) \mu \alpha \frac{N}{K}\right)
\]

(60)

B.4.3 Number of 1-hop and 2-hop neighbors in partial-i.i.d.

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

\[
\frac{N}{K} + \frac{K-1}{K} N \left(1 - \alpha \frac{N}{K} (1 - \mu)^{\frac{N}{K}}\right) \approx \frac{N}{K} + \frac{K-1}{K} N \alpha \frac{N}{K^2} (1 - \mu p + \mu K)
\]

\[
= \frac{N}{K} + \frac{K-1}{K} N \alpha \frac{N}{K^2} (1 - \mu p + \alpha \frac{N}{K^2} \mu K)
\]

(61)

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

\[
\frac{N}{K} \left(1 + (K-1) (\alpha \frac{N}{K^2} (1 - \mu p + \alpha \frac{N}{K^2} \mu K)\right)
\]

(62)

B.5 Data Distribution with Labels

For simplicity, we assume the number of classes equal to the number of clients \(K\).

For global label distribution, we have

\[
Y^T Y = N \text{diag} \left(\frac{1}{K}, ..., \frac{1}{K}\right)
\]

(63)

B.5.1 IID

For local gradient without communication and global gradient,

\[
Y^T_k Y_k = N_k \text{diag} \left(\frac{1}{K}, ..., \frac{1}{K}\right)
\]

(64)
Let $N_k = \frac{N}{K}$, we have
\[
\| \frac{\partial f_k}{\partial W} - \frac{\partial f}{\partial W} \| \leq \lambda \| K Y_k^T Y_k B Y_k^T Y_k B Y_k^T Y_k B Y_k^T Y_k - Y Y^T Y B Y^T Y B Y^T Y B Y^T Y \|
\]
\[
= \lambda \| (K (N_k)^5 - N^5) \text{diag} \left( \frac{1}{K}, \ldots, \frac{1}{K} \right) B^4 \|
\]
\[
= \lambda (1 - K \frac{(N_k)^5}{N^5}) N^5 \| \text{diag} \left( \frac{1}{K}, \ldots, \frac{1}{K} \right) B^4 \|
\]
\[
= \lambda (1 - K \frac{(N_k)^5}{N^5}) N^5 \| B^4 \|
\]
\[\text{(65)}\]

For local gradient with 1-hop communication and global gradient,
\[
\bar{Y}_k Y_k^T = \| N_k \| \text{diag} \left( \frac{1}{K}, \ldots, \frac{1}{K} \right)
\]
\[\text{(66)}\]

\[
\| \frac{\partial f_k}{\partial W} - \frac{\partial f}{\partial W} \| \leq \lambda \| K \bar{Y}_k^T \bar{Y}_k B \bar{Y}_k^T \bar{Y}_k B \bar{Y}_k^T \bar{Y}_k B \bar{Y}_k^T \bar{Y}_k - Y Y^T Y B Y^T Y B Y^T Y B Y^T Y \|
\]
\[
= \lambda \| (K (N_k)^5 |N_k|^2 - N^5) \text{diag} \left( \frac{1}{K}, \ldots, \frac{1}{K} \right) B^4 \|
\]
\[
= \lambda \| (K \frac{N^3}{K^3} + \frac{K-1}{K} N (\alpha \frac{N}{K} + \mu \alpha (K-1) N \frac{K^2}{K^2}))^2 - N^5 \| \text{diag} \left( \frac{1}{K}, \ldots, \frac{1}{K} \right) B^4 \|
\]
\[
= \lambda \| (K \frac{N^5}{K^5} (1 + (K-1)(\alpha \frac{N}{K} + \mu \alpha (K-1) N \frac{K^2}{K^2}))^2 - N^5 \| \text{diag} \left( \frac{1}{K}, \ldots, \frac{1}{K} \right) B^4 \|
\]
\[
= \lambda (1 - \frac{1}{K^4} (1 + (K-1)(\alpha \frac{N}{K} + \mu \alpha (K-1) N \frac{K^2}{K^2}))^2 N^5 \| B^4 \|
\]
\[\text{(67)}\]

For local gradient with 2-hop communication and global gradient,
\[
\bar{Y}_k \bar{Y}_k^T = \| N_k^2 \| \text{diag} \left( \frac{1}{K}, \ldots, \frac{1}{K} \right)
\]
\[\text{(68)}\]

\[
\| \frac{\partial f_k}{\partial W} - \frac{\partial f}{\partial W} \| \leq \lambda \| K \bar{Y}_k^T \bar{Y}_k B \bar{Y}_k^T \bar{Y}_k B \bar{Y}_k^T \bar{Y}_k B \bar{Y}_k^T \bar{Y}_k - Y Y^T Y B Y^T Y B Y^T Y B Y^T Y \|
\]
\[
= \lambda \| (K (N_k)^5 |N_k|^2 - N^5) \text{diag} \left( \frac{1}{K}, \ldots, \frac{1}{K} \right) B^4 \|
\]
\[
= \lambda \| (K \frac{N^3}{K^3} + \frac{K-1}{K} N (\alpha \frac{N}{K} + \mu \alpha (K-1) N \frac{K^2}{K^2}))^2 ((\frac{N}{K} + \frac{K-1}{K} N (\alpha \frac{N}{K} + \mu \alpha (K-1) N \frac{K^2}{K^2}))^2 - N^5) \|
\]
\[
= \lambda \| (K \frac{N^5}{K^5} (1 + (K-1)(\alpha \frac{N}{K} + \mu \alpha (K-1) N \frac{K^2}{K^2}))^2 - N^5) \| \text{diag} \left( \frac{1}{K}, \ldots, \frac{1}{K} \right) B^4 \|
\]
\[
= \lambda (1 - \frac{1}{K^4} (1 + (K-1)(\alpha \frac{N}{K} + \mu \alpha (K-1) N \frac{K^2}{K^2}))^2 N^5 \| B^4 \|
\]
\[\text{(69)}\]

### B.5.2 Non-IID and Partial IID

Different from i.i.d, we need to take the difference between the local distribution and global distribution. The local distribution of Non-IID is
\[
Y_k Y_k^T = \frac{N}{K} \text{diag}(1, \ldots, 0).
\]
\[\text{(70)}\]
FedGCN: Convergence and Communication Tradeoffs in Federated Training of Graph Convolutional Networks

The local label distribution is
\[ \text{diag}(1, ..., 0). \]

For simplicity, we use \( \sigma \) to represent the difference between local distribution, \( \text{diag}(1, ..., 0) \), and global distribution, \( \text{diag}(\frac{1}{K}, ..., \frac{1}{K}) \). In partial IID, we can then use \( p\sigma \).

C 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 \( \tau \). FedGCN only requires low communication cost at the initial step. Table ?? shows the communication cost of FedGCN under different data distributions.

| Methods       | 1-hop          | L-hop | BDS-GCN         |
|---------------|----------------|-------|-----------------|
| Generic Graph | \( C_1 + N\bar{d} \) | \( C_1 + \sum_{k=1}^{K}|\mathcal{N}_k|-1|d \) | \( LT\tau\rho d \sum_{k=1}^{K} |\mathcal{N}_k^V/V_k| \) |

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

C.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 \( i \).

C.1.1 Non-i.i.d.

Possibility that there is no connected node in client \( j \) for node \( i \) is
\[ (1 - \mu\alpha)\bar{\pi}. \]

Possibility that there is at least one connected node in client \( j \) for node \( i \) is
\[ 1 - (1 - \mu\alpha)\bar{\pi}. \]

Number of clients that node \( i \) needs to communicate with is
\[ 1 + (K - 1)(1 - (1 - \mu\alpha)\bar{\pi}). \]

The communication cost of \( N \) nodes is
\[ N(1 + (K - 1)(1 - (1 - \mu\alpha)\bar{\pi}))d. \]

1-order Approximation To better understanding the communication cost, we can expand the form to provide 1-order approximation
\[ (1 - \mu\alpha)\bar{\pi} \approx 1 - \mu\alpha \frac{N}{K} \]

Possibility that there is no connected node in client \( j \) for node \( i \) is
\[ 1 - (1 - \mu\alpha)\bar{\pi} \approx 1 - 1 + \mu\alpha \frac{N}{K} = \mu\alpha \frac{N}{K}. \]

The number of clients that node \( i \) needs to communicate with is then
\[ 1 + (K - 1)(1 - \mu\alpha)\bar{\pi}) \approx 1 + (K - 1)\mu\alpha \frac{N}{K}. \]
C.1.2 i.i.d.

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

$$\left(1 - \alpha\right)^{\frac{N}{K^2}} \left(1 - \mu \alpha\right)^{\frac{(K-1)N}{K^2}}. \quad (79)$$

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

$$1 - \left(1 - \alpha\right)^{\frac{N}{K^2}} \left(1 - \mu \alpha\right)^{\frac{(K-1)N}{K^2}}. \quad (80)$$

Number of clients that node $i$ needs to communicate with is

$$1 + (K - 1)(1 - \left(1 - \alpha\right)^{\frac{N}{K^2}} \left(1 - \mu \alpha\right)^{\frac{(K-1)N}{K^2}}). \quad (81)$$

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

$$N(1 + (K - 1)(1 - \left(1 - \alpha\right)^{\frac{N}{K^2}} \left(1 - \mu \alpha\right)^{\frac{(K-1)N}{K^2}}))d. \quad (82)$$

1-order Approximation

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

$$1 - \left(1 - \alpha\right)^{\frac{N}{K^2}} \left(1 - \mu \alpha\right)^{\frac{(K-1)N}{K^2}} \approx 1 - \frac{N(1 - \alpha)(K - 1)N}{K^2} + \frac{N \mu \alpha (K - 1)N}{K^2}. \quad (83)$$

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

$$1 + (K - 1)(\alpha \frac{N}{K^2} + \mu \alpha \frac{(K - 1)N}{K^2})). \quad (84)$$

C.1.3 Partial-i.i.d.

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

$$N(1 + (K - 1)(1 - \left(1 - \alpha\right)^{\frac{Np}{K^2}} \left(1 - \mu \alpha\right)^{\frac{N(1-p)}{K^2}}))d. \quad (85)$$

1-order Approximation
The number of clients that node $i$ needs to communicate with is then

$$1 - (1 - \alpha)^{N(1-p)} \approx 1 - (1 - \alpha)^{Np} (1 - \mu \alpha)^{N(K-p)} \approx 1 - (1 - \alpha^\frac{Np}{K^2} (1 - \mu \alpha)^\frac{N(K-p)}{K^2})$$

$$= 1 - (1 - \alpha)^{Np} (1 - \mu \alpha)^{N(K-p)} + \alpha Np \frac{Np}{K^2} \mu \alpha - \alpha Np \frac{Np}{K^2} \mu \alpha$$

$$\approx \alpha Np \frac{Np}{K^2} + \mu \alpha - \alpha Np \frac{Np}{K^2} \mu \alpha$$

$$= \alpha Np \frac{Np}{K^2} + \mu \alpha$$

$$= \alpha Np \frac{Np}{K^2} (1 - \mu) + \mu \alpha$$

The communication cost of all nodes is then

$$N(1 + (K - 1)\alpha \frac{N}{K^2} ((1 - \mu)p + \mu K))d. = ((1 - \mu)p + \mu K) \frac{\alpha N (K-1)}{K^2} + 1)Nd.$$}

$$= ((1 - \mu)p + \mu K) \frac{\alpha N (K-1)}{K} + 1)Nd.$$}  

(86)

$$= \frac{(1 - \mu)p + \mu K}{K} + \frac{\mu N (K-1)}{K} + 1)Nd.$$}  

(87)

### C.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,

$$\sum_{k=1}^{K} |V_k|d = Nd.$$}  

(88)

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,

$$\sum_{k=1}^{K} |\mathcal{N}_{V_k}|d$$}  

(89)

The number of neighbors in partial i.i.d for client $k$

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

$$= N \frac{K - 1}{K} N((1 - \mu)p + \mu K)$$

$$= N \frac{K - 1}{K} N((1 - \mu)p + \mu K)$$

(90)

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

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

(91)
FedGCN: Convergence and Communication Tradeoffs in Federated Training of Graph Convolutional Networks

| Scheme          | BGV | CKKS          |
|-----------------|-----|---------------|
| ring dimension  | 4096| 4096          |
| security level  | HEStd, 128, classic | HEStd_128, classic |
| multi depth     | 1   | 1             |
| scale factor bits | \   | 30            |

Table 6. FHE Scheme Parameter Configuration On PALISADE. Multi depth is configured to be 1 for optimal (minimum) maximum possible multiplicative depth in our evaluation.

| Array Size | Plaintext (Bool) | BGV (Boolean Packing) | Plaintext (Long & Double) | CKKS | BGV |
|------------|------------------|-----------------------|---------------------------|------|-----|
| 1k         | 1 kB             | 398 kB                | 8 kB                      | 266 kB | 398 kB |
| 10k        | 10 kB            | 398 kB                | 80 kB                     | 798 kB | 1 MB |
| 100k       | 100 kB           | 398 kB                | 800 kB                    | 7 MB  | 12 MB |
| 1M         | 1 MB             | 2 MB                  | 8 MB                      | 70 MB | 119 MB |
| 100M       | 100 MB           | 160 MB                | 800 MB                    | 7 GB  | 12 GB |
| 1B         | 1 GB             | 2 GB                  | 8 GB                      | 70 GB | 119 GB |

Table 7. Communication Cost After Encryption. Plaintext files are numpy arrays with pickle and ciphertext files are generated under BGV and CKKS.

The communication cost is then

\[
(N + (K - 1)N(\alpha \frac{N}{K^2}(1 - \mu)p + \alpha \frac{N}{K^2} \mu K))d = (1 + (K - 1)(\alpha \frac{N}{K^2}(1 - \mu)p + \alpha \frac{N}{K} \mu))Nd
\]

\[
= (1 + (K - 1)\alpha \frac{N}{K^2}(1 - \mu)p + \mu\alpha(K - 1)\frac{N}{K})Nd
\]

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,

\[
\sum_{k=1}^{K} |N_{V_{i_k}}^{L-1}|d.
\]

D Fully Homomorphic Encryption Microbenchmarking

We implement our FHE module using the FHE library PALISADE (v1.10.5) PALISADE (2020) with the cryptocontext parameters configuration as in Table 6. In our paper, we evaluate two FHE schemes for integer and real number respectively, i.e., an integer scheme Brakerski-Gentry-Vaikuntanathan (BGV) Brakerski et al. (2014) and a real-number scheme Cheon-Kim-Kim-Song (CKKS) Cheon et al. (2017).

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 7. In general, secure computation using FHE 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.

E Additional Experimental Results

E.1 Convergence time and communication cost on SBM model.

As shown in Figure 10, FedGCN with 2-hop approximation converges faster but requires more communication cost, which is consistent with the experiments in real datasets.
FedGCN: Convergence and Communication Tradeoffs in Federated Training of Graph Convolutional Networks

Figure 10. Convergence time and communication cost of methods on data distribution with Stochastic Block Model.

| Method        | Cora  | Citeseer | Pubmed |
|---------------|-------|----------|--------|
|               | i.i.d. | non-i.i.d. | i.i.d.       | non-i.i.d. | i.i.d.         | non-i.i.d. |
| Central/Distributed | 0.8013 | 0.8013 | 0.8013 | 0.8023 | 0.6523 | 0.6523 | 0.6745 |
| FedGCN(0-hop)  | 0.6915 | 0.7475 | 0.8163 | 0.6523 | 0.6629 | 0.6805 | 0.7306 |
| BDS-GCN       | 0.7365 | 0.7791 | 0.8079 | 0.6580 | 0.6422 | 0.6830 | 0.7739 |
| FedGCN(1-hop) | 0.8001 | 0.8030 | 0.8097 | 0.6650 | 0.6776 | 0.6836 | 0.7762 |

Table 8. Test accuracy(upper) and standard deviations(bottom) for i.i.d., non-i.i.d., and 50% i.i.d. data. FedGCN performs best on i.i.d. data, where FedGCN (0-hop) has the most information loss. As the data becomes more non-i.i.d., FedGCN (0-hop) performs better, though the variance between local and global loss gradients increases, which can impede its convergence. FedGCN (1-hop) balances the two and almost always outperforms BDS-GCN.

E.2 Test Accuracy on real datasets

Table 8 shows the final test accuracy of each training method on all three datasets. FedGCN (1-hop) generally performs well, outperforming BDS-GCN in all cases except non-i.i.d. data on Pubmed. Thus, FedGCN (1-hop) effectively balances fast convergence with limited communication, similar to our SBM validation in Figure 10. On i.i.d. data, as we would expect, FedGCN performs the best on Cora and a close second-best on Citeseer and Pubmed. FedGCN (0-hop) performs poorly due to information loss, while FedGCN (1-hop) performs well on all datasets. BDS-GCN performs poorly on Cora and Citeseer due to its having to sample from many cross-client edges.

Under partially i.i.d. data, FedGCN has the best performance for Pubmed, and is very close to the best performance for Cora and Citeseer. FedGCN (1-hop) continues to perform well on all three datasets, due to its ability to partially recover the information loss. BDS-GCN performs poorly on Citeseer and Pubmed, perhaps due to the larger size of these graphs and randomness due to information sampling. BDS-GCN does, however, perform well on Pubmed in the non-i.i.d. setting: randomly sampling cross-client information may add more diversity to the training data, helping it generalize. In the non-i.i.d. setting, FedGCN (0-hop) outperforms or is close to the performance of FedGCN (1-hop), indicating that additional communication does not meaningfully improve the model. FedGCN performs poorly, perhaps due to the fact that cross-client edges would terminate at nodes of different classes, hindering the model accuracy.
FedGCN uses orders of magnitude less communication than BDS-GCN, while achieving higher test accuracy.

E.3 Test accuracy vs. communication cost

E.4 Accuracy with different fractions of i.i.d. data

Figure 12 shows the validation accuracy of FedGCN (0-hop) during the training process for different fractions of i.i.d. data. We observe that, as the data becomes more i.i.d., the algorithm converges faster. Data that is more i.i.d. will experience more information loss, and thus it will be more difficult for this model to converge. When the fraction of i.i.d. data is above 0.8, however, more i.i.d. data leads to a higher final accuracy, likely due to the fact that federated learning is not guaranteed to converge if the data is i.i.d., because of discrepancies between clients’ local updates.

Figure 12. Validation accuracy of 0-hop Approximation on data distribution with Cora Dataset.

F 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.