Graph Federated Learning with Hidden Representation Sharing

Shuang Wu*
UCLA
Los Angeles, CA, USA
shuangwu222@ucla.edu

Mingxuan Zhang†
Purdue University
West Lafayette, IN, USA
zhan3692@purdue.edu

Yuantong Li
UCLA
Los Angeles, CA, USA
yuantongli@ucla.edu

Carl Yang
Emory University
Atlanta, GA, USA
j.carlyang@emory.edu

Pan Li
Purdue University
West Lafayette, IN, USA
panli@purdue.edu

ABSTRACT

Learning on Graphs (LoG) is widely used in multi-client systems when each client has insufficient local data, and multiple clients have to share their raw data to learn a model of good quality. One scenario is to recommend items to clients with limited historical data and sharing similar preferences with other clients in a social network. On the other hand, due to the increasing demands for the protection of clients' data privacy, Federated Learning (FL) has been widely adopted: FL requires models to be trained in a multi-client system and restricts sharing of raw data among clients. The underlying potential data-sharing conflict between LoG and FL is under-explored and how to benefit from both sides is a promising problem. In this work, we first formulate the Graph Federated Learning (GFL) problem that unifies LoG and FL in multi-client systems and then propose sharing hidden representation instead of the raw data of neighbors to protect data privacy as a solution. To overcome the biased gradient problem in GFL, we provide a gradient estimation method and its convergence analysis under the non-convex objective. In experiments, we evaluate our method in classification tasks on graphs. Our experiment shows a good match between our theory and the practice.

ACM Reference Format:
Shuang Wu, Mingxuan Zhang, Yuantong Li, Carl Yang, and Pan Li. 2022. Graph Federated Learning with Hidden Representation Sharing. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (ACM CIKM Workshop'2022).* ACM, Atlanta, GA, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Learning on Graphs (LoG) in multi-client systems has extensive applications such as Graph Neural Networks (GNNs) for recommendation [6, 7, 35, 36], finance [24, 34], and traffic [3, 39]. The key to the success of LoG is sharing local raw data between clients. For example, when recommending items to users with insufficient local data, data sharing from their friends with similar preferences in a social network can improve the performance of recommendation models. On the other hand, Federated Learning (FL) is widely explored due to its protection of data privacy, especially in medical fields [38], mobile device fields [10, 23], and Internet of Things (IoT) fields [25]. In FL, models are trained without data sharing among clients to protect clients' local data privacy. As a consequence, combining FL and LoG in multi-client systems faces a fundamental conflict in data sharing.

As we know, considerable works are combing Federated Learning and Graph Machine Learning. One attractive research line is using FL to train GNNs [33, 43]. In addition, [27, 36] use FL to train GNN-based models to address specific real-world applications. [11, 42] summarises current efforts on FL over graphs, including the above literature. However, most current works did not utilize the network of clients in the system and failed to protect the privacy of the nodes in the network. In other words, previous literature never models FL clients as nodes in GNNs on multi-client systems. Besides, all these works are application-oriented without a theoretical guarantee. Therefore, fundamental data sharing conflict remains unsolved.

Such significant conflict motivates our investigation of the construction of Graph Federated Learning (GFL) in multi-client systems: **Can we formulate a GFL framework to address the data sharing conflict, paired with theoretical and empirical supports?** We aim to deliver a generic framework of GFL. Our work focuses on the centralized federated learning setting while data collected by clients are Non-IID distributed.

Contributions. We formulate the GFL problem for a graph-based model in multi-client systems. To address the data sharing conflict, we propose an FL solution with the hidden representation sharing technique, which only requires the sharing of hidden representations rather than the raw data from the neighbors to protect data privacy on multi-client systems. A technical challenge arises since the hidden representations are only exchanged during communication between clients and the central server, making unbiased gradient estimation becomes impractical. As a remedy, we provide a practical gradient estimation method. Moreover, a convergence analysis with non-convex objectives of the proposed algorithm is provided. To the best of our knowledge, this is the first theoretical analysis for FL with a graph-based model. We propose `GFL-APPNP` and empirically evaluate the proposed method for several classification tasks on graphs, including deterministic node classification, stochastic node classification, and supervised classification. Our experiments show that the proposed method converges and achieves competitive performance. Additionally, the

*Contributed equally to this research. Corresponding author.
†Contributed equally to this research.

ACM CIKM Workshop'2022, October, 2022, Atlanta, GA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

results provide a good match between our theory and practice. The contributions in this paper are summarized as follows:

- Formulate the GFL problem to model FL clients as nodes in LoG on multi-client systems.
- Propose FL solution with hidden representation sharing for GFL problem to resolve data sharing conflict.
- Provide theoretical non-convex convergence analysis for GFL.
- Propose GFL-APPNP and empirically show the proposed algorithm is valid and has competitive performance on classification tasks.

2 RELATED WORKS

Federated Learning for GNNs. How to utilize the FL technique to train GNNs is an interesting topic that attracts lots of attention from researchers. For instance, [33] focuses on graph semi-supervised learning via meta-learning and handles testing nodes with new label domains as well as leverages unlabeled data. [43] proposes federated learning to train GNNs by dividing a large graph into subgraphs. [37] considers an FL solution to train GNNs for the entire graph classification. [12] proposes decentralized periodic SGD to solve the serverless Federated Multi-Task Learning for GNNs. [27] Proposes a GNN-based federated learning architecture for spatio-temporal data modeling. [36] puts forward a decentralized federated framework for privacy-preserving GNN-based recommendations. However, [11, 33, 37, 43] assume each client has its own graphs and [27, 36] use federated learning to train GNN-based model. None of these works is federated learning to train GNNs on multi-client systems with the protection of node-level privacy, which is addressed by our work.

Personalized Federated Learning. The conventional FL approach faces a fundamental challenge of poor performance on highly heterogeneous clients. Previous works [21, 22] provided solutions to tackling Non-IID data across clients. Recently, inspired by personalization, research on personalized federated learning has developed rapidly [5, 26, 31]. Particularly, personalization with graph structure to tackle heterogeneity in FL is highly related to our work. For example, [29] proposes MOCHA which uses a graph-type regularization to control the parameters and perform a prime-dual framework, and [9] provides a similar regularizer to the multitask learning. [31] considers an implicit model where personalized parameters come from a Moreau envelope and this idea recently has got generalized to graph Laplacian regularization [4]. [19, 20] assumes that there is a common parameter shared across the network when each node of the graph is viewed as a federated learning client that generates independent data. All these works are model-level personalization based on graphs such as graph regularization while LoG encourages data-level sharing.

Notations. Let $[n]$ be the set $\{1, 2, ..., n\}$. Vectors are assumed to be column vectors. $\mathbf{1}$ is a vector with all ones. $\boldsymbol{I}$ is the identity matrix with appropriate dimension. $\|\cdot\|$ is assumed to be the 2-norm. For a matrix $\boldsymbol{A}$, $\lambda_{max}(\boldsymbol{A})$ is the maximum eigenvalue of $\boldsymbol{A}$ and $\boldsymbol{A}^{\dagger}$ is the Moore–Penrose inverse of $\boldsymbol{A}$. $O(\cdot)$ is the big-O notation.

3 GRAPH FEDERATED LEARNING

3.1 Preliminaries

Federated Learning. In typical FL, multiple clients collectively solve a global task. Our work focuses on the centralized setting with a central server, and we consider the following consensus optimization problem:

$$\min_{\boldsymbol{W}} \ F(\boldsymbol{W}) := \frac{1}{N}\sum_{k=1}^{N} F_k(\boldsymbol{W}) \tag{1}$$

where N is the number of clients and $\boldsymbol{W}$ is the model parameter. $F(\boldsymbol{W})$ is the global loss function and $F_k(\boldsymbol{W})$ is the local loss function. For client k, it has access only to its local data and conducts local update $\boldsymbol{W}_k^{t+1} \leftarrow \boldsymbol{W}_k^t - \eta \boldsymbol{g}_k^t$ where $\boldsymbol{g}_k^t := \nabla \hat{F}_k(\boldsymbol{W}_k^t)$ is the stochastic gradient estimator of $\nabla F_k(\boldsymbol{W}_k^t)$ and η is the learning rate. Throughout our work, ∇ is gradient w.r.t model parameter $\boldsymbol{W}$. Denote I as the number of local updates between two communication rounds. During the FL process, after I steps of local update, the central server aggregates the latest models from clients according to FedAvg [17]: $\bar{\boldsymbol{W}}^t = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{W}_k^t$.

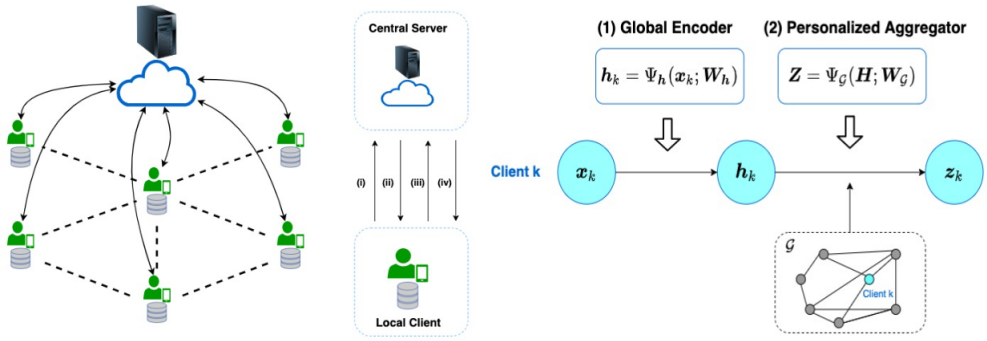
Statistical Heterogeneity. The goal of FL is to minimize the global loss on the average data distribution across clients, as shown in Eq.(1). However, in most substantial applications of FL, clients collect data in a Non-IID distributed manner, leading to a fundamental statistical heterogeneity/shift problem in FL [14, 21]. [5, 18] have suggested quantification of statistical heterogeneity. In this paper, we use the term "level of statistical heterogeneity" to describe how large the statistical shift is across clients.

3.2 GFL Problem Formulation

The topological structure which describes the Non-IID relationship among clients' distributions is an undirected graph denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of N clients and $\mathcal{E}$ is a set of edges. The adjacency matrix of $\mathcal{G}$ is denoted as $\boldsymbol{A} \in \{0, 1\}^{N\times N}$. Throughout this paper, nodes in graph $\mathcal{G}$ are referred to as clients. Furthermore, denote $\Xi_{\mathcal{G}} = (\boldsymbol{X}, \boldsymbol{Y}) \in \boldsymbol{R}^{N\times(d+c)}$ as the data matrix where $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ is the feature matrix with the number of features d and $\boldsymbol{Y} \in \mathbb{R}^{N\times c}$ is the label matrix with the dimension of label c. To formulate the GFL problem generally, consider $\Xi_{\mathcal{G}}$ as a random matrix from a distribution $\mathcal{D}_{\mathcal{G}}$ which depends on $\mathcal{G}$, that is, $\Xi_{\mathcal{G}} \sim \mathcal{D}_{\mathcal{G}}$. More specifically, we define the k-th row vector of $\Xi_{\mathcal{G}}$ as $\boldsymbol{\xi}_k := (\boldsymbol{x}_k, \boldsymbol{y}_k)$ where $\boldsymbol{x}_k$ is the feature vector and $\boldsymbol{y}_k$ is a vector of labels in client k. Thus $\Xi_{\mathcal{G}}$ is the random data matrix whose rows are correlated and the relationship between $\boldsymbol{\xi}_k$ is described by graph $\mathcal{G}$. Here we assume that graph structure $\mathcal{G}$ is deterministic. With these notations, the GFL problem is defined as:

$$\min_{\boldsymbol{W}} \ \frac{1}{N}\sum_{k=1}^{N} F_k(\boldsymbol{W}), \text{where } F_k(\boldsymbol{W}) := \mathbb{E}[f_k(\boldsymbol{W}; \Xi_{\mathcal{G}})], \tag{2}$$

and $f_k(\boldsymbol{W}; \Xi_{\mathcal{G}})$ is the local loss after observing data matrix $\Xi_{\mathcal{G}}$, indicating the local objective function of client k depends not only on the data collected by the k-th client but also the data from other clients. This is the key difference between GFL and conventional FL. As our discussion in §1, this crucial difference induces the data-sharing conflict. Therefore, we propose the following hidden



(a) Communications in GFL (b) Illustration of proposed model

Figure 1: (a) four steps in each communication round: (i) Uploading models. (ii) Broadcasting aggregated model. (iii) Uploading hidden representations. (iv) Broadcasting hidden representations. (b) global encoder Ψ_h is shared since the same task is shared among clients. This is why FedAvg can be utilized in this multi-clients system. The personalized aggregator $\Psi_{\mathcal{G}}$ accounts for the statistical heterogeneity across clients.

representation sharing method to address the challenge of data-sharing conflict in the GFL problem.

3.3 Hidden Representation Sharing

Our proposal is using hidden representation. The hidden representations are allowed to be shared across clients in network $\mathcal{G}$, and a neighborhood aggregator is applied to these hidden representations of all nodes. For client k, define its hidden representation $\boldsymbol{h}_k$ as follow

$$\boldsymbol{h}_k = \Psi_h(\boldsymbol{x}_k; \boldsymbol{W}_h) \tag{3}$$

where $\Psi_h(\cdot; \boldsymbol{W}_h)$ is a hidden encoder such as the multi-layer perceptron (MLP) parametrized by $\boldsymbol{W}_h$. The hidden representation matrix is denoted as $\boldsymbol{H} \in \mathbb{R}^{N \times d_h}$ where the k-th row vector of $\boldsymbol{H}$ is $\boldsymbol{h}_k$ and d_h is the dimension of the hidden representations. Graph representations are defined by neighborhood representation aggregation of $\boldsymbol{H}$:

$$\boldsymbol{Z} = \Psi_{\mathcal{G}}(\boldsymbol{H}; \boldsymbol{W}_{\mathcal{G}}) \tag{4}$$

where $\Psi_{\mathcal{G}}(\cdot; \boldsymbol{W}_{\mathcal{G}})$ is a neighborhood aggregator parametrized by $\boldsymbol{W}_{\mathcal{G}}$. $\boldsymbol{Z} \in \mathbb{R}^{N \times d_z}$ is the graph representation matrix whose k-th row vector is denoted as $\boldsymbol{z}_k$ and d_z is the dimension of the graph representations. In most classification tasks, $d_z = c$. Model parameter $\boldsymbol{W} = (\boldsymbol{W}_h, \boldsymbol{W}_{\mathcal{G}})^\top$ is the concatenate of $\boldsymbol{W}_h$ and $\boldsymbol{W}_{\mathcal{G}}$. With these privacy-preserving representations, loss function for the corresponding graph federated optimization can be express as, $f_k(\boldsymbol{W}; \Xi_{\mathcal{G}}) := \ell(\boldsymbol{y}_k, \boldsymbol{z}_k)$ where ℓ is the pre-specified loss function such as cross entropy for classification task.

Remark 1. *The explication in Figure 1 shows that the hidden encoder is a global model which facilitates the involvement of FL, while the neighborhood aggregator is the personalized model which accounts for statistical heterogeneity. Intuitively, Ψ_h contributes to privacy protection and representation extraction. Meanwhile, $\Psi_{\mathcal{G}}$ serves as modeling the heterogeneity using graph $\mathcal{G}$. Note that if we set $\Psi_{\mathcal{G}}$ as the identity mapping (ignore the graph information), our solution reduces to the conventional FL solution to learn a global model Ψ_h. In addition, when $\mathcal{G}$ does not fully capture the relationship across clients, $\boldsymbol{W}_{\mathcal{G}}$ serves as weights for adjusting the neighborhood aggregation level using $\mathcal{G}$.*

3.4 Gradient Estimation

In practice, to solve the GFL problem by gradient-based methods, the unbiased stochastic gradient $\nabla f_k(\boldsymbol{W}; \Xi_{\mathcal{G}})$ of client k depends on data from all nodes in the network $\mathcal{G}$ ($\mathbb{E}[f_k(\boldsymbol{W}; \Xi_{\mathcal{G}})] = F_k(\boldsymbol{W})$). However, since FL restricts the data-sharing, $\nabla f_k(\boldsymbol{W}; \Xi_{\mathcal{G}})$ is inaccessible. Another estimation of $\nabla F_k(\boldsymbol{W})$ for local updates must be raised. In the proposed hidden representation sharing method, local information is exchanged as a function of $\{\boldsymbol{h}_j, \nabla \boldsymbol{h}_j\}_{j=1}^N$ during the interactions between clients and the central server. In other words, if the client k can access $\{\boldsymbol{h}_j, \nabla \boldsymbol{h}_j\}_{j=1}^N$, the unbiased estimator $\nabla f_k(\boldsymbol{W}; \Xi_{\mathcal{G}})$ is accessible. Formally, with the shared hidden representations, $\nabla f_k(\boldsymbol{W}; \Xi_{\mathcal{G}})$ can be expressed as a function of hidden representations: $\nabla f_k(\boldsymbol{W}; \Xi_{\mathcal{G}}) = \phi_k(\boldsymbol{h}_1, ..., \boldsymbol{h}_N)$. Note that $\nabla \boldsymbol{h}_j$ is also a function of $\boldsymbol{h}_j$ and we consider the case that estimating $\nabla \boldsymbol{h}_j$ is completely based on an estimator of $\boldsymbol{h}_j$. Furthermore, define $\hat{\boldsymbol{h}}_{j \to k}$ as the estimation of the hidden representation $\boldsymbol{h}_j$ for client k. Then the biased estimator of $\nabla F_k(\boldsymbol{W})$ is,

$$\nabla \hat{f}_k(\boldsymbol{W}; \xi_k) = \phi_k(\hat{\boldsymbol{h}}_{1 \to k}, ..., \hat{\boldsymbol{h}}_{N \to k}), \forall k \in [N]. \tag{5}$$

The strategy to design estimator $\hat{\boldsymbol{h}}_{j \to k}$ depends on the concrete scenario. In §5, we provide gradient compensation strategy with theoretical analysis in Appendix A.1 and the empirical results of this biased estimation strategy is provided in §6. In practice, the estimator of $\nabla F_k(\boldsymbol{W})$ is the batch mean of biased stochastic gradients. Formally, suppose $\mathcal{B}_k := \{\xi_{k,s}\}_{s=1}^{|\mathcal{B}_k|}$ is the mini-batch with batch size $|\mathcal{B}_k|$ for some local update in client k. $\nabla \hat{f}_k(\boldsymbol{W}; \xi_{k,s})$ is the estimated gradient which depends on the example $\xi_{k,s}$. Batch mean of biased stochastic gradients is defined as $\nabla \hat{F}_k(\boldsymbol{W}) := \frac{1}{|\mathcal{B}_k|} \sum_{s \in \mathcal{B}_k} \nabla \hat{f}_k(\boldsymbol{W}; \xi_{k,s})$.

Privacy in GFL. FL and GFL require the protection of node-level privacy: client can not share their own collected data with both other clients and the central server directly. However, directly sharing $\{\boldsymbol{h}_j, \nabla \boldsymbol{h}_j\}_{j=1}^N$ raises the concern about raw data recovery

by untrustworthy clients or the central server. Our proposed solution does not violate node-level privacy even though we allow sharing hidden representations and the corresponding gradients during the communication between clients and the central server. By using the personalized neighborhood aggregator, clients will not receive $\{\boldsymbol{h}_j, \nabla \boldsymbol{h}_j\}_{j=1}^N$ directly, making the raw data recovery infeasible. In addition, the concern about the unreliable server can be addressed by better design of the central server or adding noise to the gradients.

3.5 Graph Federated Learning Procedure

A framework of communications in GFL with hidden representation sharing is described in Figure 1. An concrete example is Algorithm 1 introduced in § 5.2. Steps at each communication round are:

(1) **Uploading Models:** Clients parallelly upload the latest models to the central server.
(2) **Centralizing Models:** Central server aggregates models by `FedAvg` and broadcasts the aggregated result.
(3) **Uploading Hidden Representations:** Clients compute estimated hidden representation and their gradient using the received aggregated model in step (2) and then parallelly upload them to the central server.
(4) **Broadcasting Hidden Representations:** Central server allocates estimated hidden representation and their gradients and broadcasts the aggregated ones to clients.
(5) **Local Updates:** Clients parallelly perform local updates for I times.

4 THEORETICAL ANALYSIS

4.1 Assumptions

Assumption 1. *(Smoothness) Local loss function F_k is differentiable and assumed to be smooth with constant ρ_f, $\forall k \in [N]$. Formally, $\forall \boldsymbol{W}, \boldsymbol{W}'$, $\exists \rho_f > 0$ such that*

$$\left\|\nabla F_k(\boldsymbol{W}) - \nabla F_k(\boldsymbol{W}')\right\| \leq \rho_f \left\|\boldsymbol{W} - \boldsymbol{W}'\right\|. \tag{6}$$

Assumption 2. *(Bound for Hidden Representation Estimation) Hidden Representation $\boldsymbol{h}_j$ of client j is estimated by $\hat{\boldsymbol{h}}_{j\to k}$ for local updates at client k. The mean squared error of estimation is bounded in the following sense: $\forall j, k \in [N]$, $\exists \sigma_j^2 > 0$ and $\sigma_H^2 := \sum_{j=1}^N \sigma_j^2$ such that,*

$$\mathbb{E}[\left\|\hat{\boldsymbol{h}}_{j\to k} - \boldsymbol{h}_j\right\|^2] \leq \sigma_j^2. \tag{7}$$

Assumption 3. *(Graph Smoothing on Gradients) Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is connected graph and $\exists \kappa^2 > 0$ such that $\forall \boldsymbol{W}$,*

$$\sum_{(i,j)\in\mathcal{E}} \left\|\nabla F_i(\boldsymbol{W}) - \nabla F_j(\boldsymbol{W})\right\|^2 \leq \kappa^2. \tag{8}$$

Assumption 4. *(Bounds for Stochastic Gradient)*
(i) (Bounded Variance) Variance of unbiased stochastic gradient $\nabla f_k(\boldsymbol{W}; \Xi_{\mathcal{G}})$ is bounded. Formally, $\exists \sigma_{\mathcal{G}}^2 > 0$ such that $\forall \boldsymbol{W}$,

$$\sum_{k=1}^N \mathbb{E}[\left\|\nabla f_k(\boldsymbol{W}; \Xi_{\mathcal{G}}) - \nabla F_k(\boldsymbol{W})\right\|^2] \leq \sigma_{\mathcal{G}}^2. \tag{9}$$

(ii) (Smoothness) Denote $\nabla f_k(\boldsymbol{W}; \Xi_{\mathcal{G}}) = \phi_k(\boldsymbol{h}_1, ..., \boldsymbol{h}_N)$. Assume for any $k \in [N]$, ϕ_k satisfies that $\forall \boldsymbol{h}_i, \boldsymbol{h}_i'$ and $i \in [N]$, $\exists \rho_\phi > 0$ such that,

$$\left\|\phi_k(\boldsymbol{h}_1, ..., \boldsymbol{h}_N) - \phi_k(\boldsymbol{h}_1', ..., \boldsymbol{h}_N')\right\| \leq \rho_\phi \big(\sum_{i=1}^N \left\|\boldsymbol{h}_i - \boldsymbol{h}_i'\right\|\big)^{1/2}. \tag{10}$$

Interpretation of Assumptions. (i) Assumption 1 is commonly assumed in the literature on nonconvex optimization and FL. (ii) Assumption 2 is the goodness of hidden representations estimation. σ_j^2 represents the estimation error of $\hat{\boldsymbol{h}}_{j\to k}$ and σ_H^2 quantifies the total estimation error. (iii) κ^2 in Assumption 3 quantifies this statistical heterogeneity among clients by considering the network structure which captures the relationship among clients' distributions. A previous work [5] shows that there is a connection between data distribution shift among clients and the gradient shift among clients. (iv) Assumption 4 ensures $\nabla f_k(\boldsymbol{W}; \Xi_{\mathcal{G}})$ satisfies two properties. First, it has a bounded variance ($\sigma_{\mathcal{G}}^2$) which is commonly presumed in previous works. The second one is a sense of smoothness of $\nabla f_k(\boldsymbol{W}; \Xi_{\mathcal{G}})$ in terms of a function of hidden representations with smoothness quantified by ρ_ϕ.

4.2 Convergence Analysis

Theorem 4.1. *Consider GFL optimization problem (2) under Assumptions 1, 2, 3 and 4. Use the federated learning procedure described in § 3. Suppose η and I satisfies $I\eta^2 < 1/13\rho_f^2$, then for all $T \geq 1$, we have*

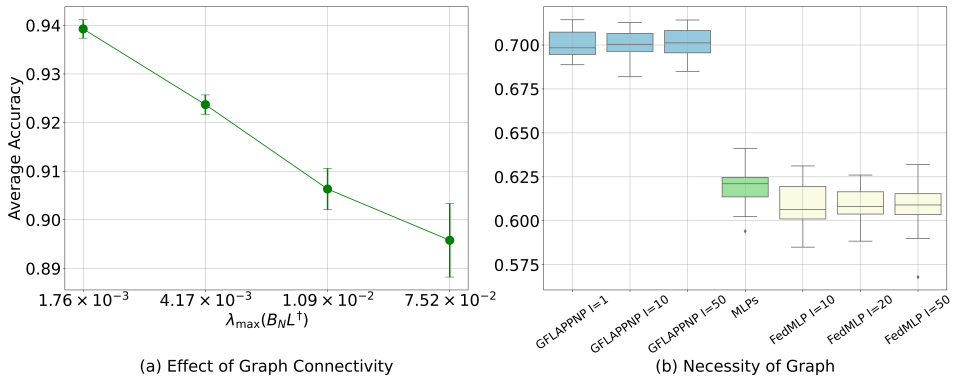
$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\left\|\nabla F(\bar{\boldsymbol{W}}^t)\right\|^2] \leq & O(\frac{1}{\eta T}) + O(\frac{I^2\eta^2\sigma_{\mathcal{G}}^2}{N}) \\ & + O(I^2\eta^2\sigma_H^2) + O(\frac{I^2\eta^2\kappa^2\lambda_{\max}(\boldsymbol{B}_N\boldsymbol{L}^\dagger)}{N})\end{aligned} \tag{11}$$

Where $\boldsymbol{B}_N := \frac{1}{N}\boldsymbol{I} - \frac{1}{N^2}\boldsymbol{1}\boldsymbol{1}^\top$ and $\boldsymbol{L}$ is the Laplacian matrix of $\mathcal{G}$.

Corollary 4.2. *Under the setting of Theorem 4.1. Suppose learning rate is chosen as $\eta = \frac{\sqrt{N}}{\sqrt{T}}$ and removing smoothness constants ρ_f and ρ_ϕ, we have*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\left\|\nabla F(\bar{\boldsymbol{W}}^t)\right\|^2] = O(\frac{1}{\sqrt{NT}}) + O(\frac{NI^2}{T}) + O(\frac{\lambda_{\max}(\boldsymbol{B}_N\boldsymbol{L}^\dagger)I^2}{T}) \tag{12}$$

Remark 2. *According to our Theorem 4.1 and Corollary 4.2, when $\sigma_H^2 = 0$, that is, ignoring the node-level privacy issue and access to the unbiased stochastic gradient, our convergence result matches the rate of the previous works [13, 30, 41] with the gradient deviation among clients is described by a graph structure. For the effect from graph structure, since $\lambda_{\max}(\boldsymbol{B}_N\boldsymbol{L}^\dagger)$ is an indication of the connectivity of graph $\mathcal{G}$ with normalized by averaged aggregation $\boldsymbol{B}_N$ (large $\lambda_{\max}(\boldsymbol{B}_N\boldsymbol{L}^\dagger)$ means a bad connectivity and high level of statistical heterogeneity), we can expect that a graph with good connectivity ensures a better performances as shown in Figure 2. This observation matches our intuition that smaller level of statistical heterogeneity in FL secures a better performance. Moreover, our Corollary 4.2 keep the linear speed up (w.r.t number of clients) when $I = 1$ [40].*



(a) Effect of Graph Connectivity (b) Necessity of Graph

Figure 2: (a) the markers represent accuracy for graphs with different connectivity measured by $\lambda_{\max}(B_N L^{\dagger})$ discussed in Remark 2. (b) box plot of average accuracy on 20 synthetic graphs over our methods and baseline models. More details are in Appendix A.2.

Algorithm 1 GFL-APPNP for Classification

Require: Initialize $\{\boldsymbol{W}_k^0\}_{k=1}^N$, η, T, I. Compute $\tilde{\boldsymbol{A}}$.
  for $t = 0, ..., T-1$ **do**
    if $t \mod I = 0$ **then**
      On Client $k \in [N]$ **Parallelly**:
        Uploads latest model $\boldsymbol{W}_k^t$.
      On Central Server:
        Broadcast $\bar{\boldsymbol{W}}^t = \frac{1}{N}\sum_{k=1}^N \boldsymbol{W}_k^t$ to all clients.
      On Client $j \in [N]$ **Parallelly**:
        Compute $\hat{\boldsymbol{h}}_j = \frac{1}{n_j}\sum_{s=1}^{n_j} \mathrm{MLP}(\boldsymbol{x}_{j,s}; \bar{\boldsymbol{W}}^t)$
        Set $\boldsymbol{W}_j^t = \bar{\boldsymbol{W}}^t$.
        Upload $\hat{\boldsymbol{h}}_j$ and $\nabla\hat{\boldsymbol{h}}_j$.
      On Central Server:
        Compute $\boldsymbol{C}_k = \sum_{j\neq k} \tilde{\boldsymbol{A}}_{kj}\hat{\boldsymbol{h}}_j, \forall k \in [N]$.
        Compute $\nabla\boldsymbol{C}_k = \sum_{j\neq k} \tilde{\boldsymbol{A}}_{kj}\nabla\hat{\boldsymbol{h}}_j, \forall k \in [N]$.
        Broadcast $\boldsymbol{C}_j$ and $\nabla\boldsymbol{C}_j$ to client j for all $j \in [N]$.
    end if
    On Client $k \in [N]$ **Parallelly**:
      Compute $\boldsymbol{h}_{k,s} = \mathrm{MLP}(\boldsymbol{x}_{k,s}; \boldsymbol{W}_k^t), \forall s \in \mathcal{B}_k$.
      Compute $\hat{\boldsymbol{y}}_{k,s} = \mathrm{Softmax}(\tilde{\boldsymbol{A}}_{kk}\boldsymbol{h}_{k,s} + \boldsymbol{C}_k), \forall s \in \mathcal{B}_k$
      Compute $\boldsymbol{g}_{k,s}^t = (\boldsymbol{y}_k - \hat{\boldsymbol{y}}_{k,s})(\tilde{\boldsymbol{A}}_{kk}\nabla\boldsymbol{h}_{k,s} + \nabla\boldsymbol{C}_k), \forall s \in \mathcal{B}_k$.
      Compute $\boldsymbol{g}_k^t = \frac{1}{|\mathcal{B}_k|}\sum_{s\in\mathcal{B}_k} \boldsymbol{g}_{k,s}^t$.
      Compute $\boldsymbol{W}_k^{t+1} \leftarrow \boldsymbol{W}_k^t - \eta\boldsymbol{g}_k^t$.
  end for

5 GFL-APPNP FOR CLASSIFICATION

5.1 GFL for Classification Tasks on Graphs

Deterministic Node Classification (DNC). Graph-based semi-supervised node classification is the most popular classification task on graphs. In this paper, we call it deterministic node classification since $\boldsymbol{\xi}_k$ for each node is deterministic with one feature vector and one label. The GFL problem can formulate this task in § 3 by assuming $\boldsymbol{\xi}_k$ is from a degenerated distribution.

Stochastic Node Classification (SNC). An extended version of the deterministic node classification is the setting where local distribution at each node is not degenerated, namely stochastic node classification. This task is a semi-supervised node classification that classifies the nodes by learning from local distributions. Similarly, this task can be formulated by the GFL problem in § 3 by assuming the randomness of $\Xi_{\mathcal{G}}$ is only from $\boldsymbol{X}$. An important real-world application for this task is the user demographic label prediction in social networks.

Supervised Classification (SC). Consider the supervised learning task on clients, which is another classification task on graphs assuming the label of client also follows a distribution. We call it supervised classification. The objective of this task is to classify the feature vectors in all clients. This task assumes the randomness of $\Xi_{\mathcal{G}}$ is from both $\boldsymbol{X}$ and $\boldsymbol{Y}$, resulting in that each client might have examples with different labels. One practical application is the patient classification in hospitals with insufficient medical records.

More details about classification tasks are provided in Appendix A.2. Moreover, our GFL setting introduced in § 3 is not only for standard supervised learning but also can be easily extended to semi-supervised client classification like DNC and SNC.

5.2 GFL-APPNP Algorithm

Approximate Personalized Propagation of Neural Predictions (APPNP) [16] is one of the state-of-the-art GNN models. With the notations and context in Section 3, APPNP has the hidden encoder Ψ_h and neighborhood aggregator $\Psi_{\mathcal{G}}$ defined as follow,

$$\boldsymbol{h}_k = \Psi_h(\boldsymbol{x}_k; \boldsymbol{W}) = \mathrm{MLP}(\boldsymbol{x}_k; \boldsymbol{W}), \tag{13a}$$

$$\boldsymbol{Z} = \sum_{i=0}^{M}(1-\alpha I\{i\neq M\})\alpha^i(\hat{\boldsymbol{D}}^{-1/2}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-1/2})^i\boldsymbol{H} = \tilde{\boldsymbol{A}}\boldsymbol{H}, \tag{13b}$$

where α is teleport probability [16] and M is the total steps for personalized propagation. $\hat{\boldsymbol{A}}$ is the adjacency matrix with self loop and $\hat{\boldsymbol{D}}$ is the degree matrix with self loop. $\tilde{\boldsymbol{A}}$ is defined as $\tilde{\boldsymbol{A}} := \sum_{i=0}^{M}(1-\alpha I\{i\neq M\})\alpha^i(\hat{\boldsymbol{D}}^{-1/2}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-1/2})^i$. It can be interpreted as

Table 1: Results on the deterministic node classification task. This table provides the results of the average test accuracy and the corresponding 95% confidence interval. "SG" represents synthetic graphs, SAGE represents GraphSAGE,.

	GFL-APPNP$_{I=10}$	GFL-APPNP$_{I=20}$	GFL-APPNP$_{I=50}$	APPNP	GCN	GAT	SAGE
SG	93.4(0.99)	93.3(0.94)	93.0(0.96)	93.2(0.92)	**95.2(0.54)**	93.3(1.03)	70.2(4.21)
SubCora	54.1(3.72)	**54.3(3.73)**	54.0(3.73)	54.2(3.69)	51.9(3.78)	47.9(3.01)	47.0(3.73)

the "adjacency matrix" after M steps random walk which shows the reachability between two nodes in the structure after propagations. Loss function ℓ in APPNP is the cross entropy loss. In APPNP, $\boldsymbol{W}_h$ discussed in Eq.(3) refers to $\boldsymbol{W}$ since the neightborhood aggregator in APPNP is not parametrized. Note that original APPNP is proposed to solve the deterministic node classification task. Denote predicted one-hot vectors as $\hat{Y} = \text{Softmax}(\boldsymbol{Z})$ where $\text{Softmax}(\cdot)$ is a row-wise softmax function. The gradients can be expressed explicitly as follow,

$$\nabla f_k(\boldsymbol{W}; \Xi_{\mathcal{G}}) = (\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k) \sum_{i=1}^{N} \tilde{\boldsymbol{A}}_{ki} \nabla \boldsymbol{h}_i, \tag{14}$$

where $\boldsymbol{y}_k$ is the one-hot vector for the true label of client k and $\hat{\boldsymbol{y}}_k$ is the predicted probability vector for the label of client k. $\tilde{\boldsymbol{A}}_{ki}$ is the element in matrix $\tilde{\boldsymbol{A}}$. Note that the hidden representation sharing contributes to two parts in the gradient for local loss f_k: one is $\hat{\boldsymbol{y}}_k$ and the other one is $\{\nabla \boldsymbol{h}_i\}_{i \neq k}$. A good property of using personalized propagation as the neighborhood aggregator is the linearity, which means

$$\hat{\boldsymbol{y}}_k = \text{Softmax}(\tilde{\boldsymbol{A}}_{kk} \boldsymbol{h}_k + \boldsymbol{C}_k), \tag{15a}$$

$$\nabla f_k(\boldsymbol{W}; \Xi_{\mathcal{G}}) = (\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k)(\tilde{\boldsymbol{A}}_{kk} \nabla \boldsymbol{h}_k + \nabla \boldsymbol{C}_k), \tag{15b}$$

where $\boldsymbol{C}_k := \sum_{i \neq k} \tilde{\boldsymbol{A}}_{ki} \boldsymbol{h}_i$ and $\nabla \boldsymbol{C}_k := \sum_{i \neq k} \tilde{\boldsymbol{A}}_{ki} \nabla \boldsymbol{h}_i$. Clearly, $\boldsymbol{C}_k$ and its gradient $\nabla \boldsymbol{C}_k$ are aggregated information for client k. Therefore, in practice, the central server only needs to broadcast $\boldsymbol{C}_k$ and its gradient to client k in the communication round, which provides a private communication since the hidden representations are not shared directly.

We propose GFL-APPNP algorithm for GFL problem on classification tasks, using hidden representation sharing. In addition, we use the latest aggregated model to compute hidden representations at each communication round as our gradient compensation strategy. As a concrete example, consider client j has n_j local feature vectors $\{\boldsymbol{x}_{j,s}\}_{s=1}^{n_j}$, suppose $t_0 < t$ is the largest multiple of I,

$$\hat{\boldsymbol{h}}_{j \to k}^{t} = \begin{cases} \Psi_h(\boldsymbol{x}_k; \boldsymbol{W}_{h,k}^{t}) & j = k \\ \frac{1}{n_j} \sum_{s=1}^{n_j} \Psi_h(\boldsymbol{x}_{j,s}; \bar{\boldsymbol{W}}_h^{t_0}) & j \neq k \end{cases}, \tag{16}$$

where $\hat{\boldsymbol{h}}_{j \to k}^{t}$ is the estimation for $\boldsymbol{h}_{j \to k}$ at time t. Our compensation strategy satisfies the guarantee discussed in Assumption 2 with additional assumptions. See Appendix A.1 for detailed discussion for gradient compensation. Summary of GFL-APPNP is described in Algorithm 1. Our proposed GFL-APPNP is a FL version for APPNP, which fulfills the FL for a GNN model. It is noteworthy that GFL-APPNP with $I = 1$ is *equivalent* to the FL for the vanilla APPNP for deterministic node classification tasks.

6 EXPERIMENTS

6.1 Deterministic Node Classification

We compare the proposed GFL-APPNP to baseline models including GCN, GAT, and GraphSAGE under the DNC setting described in §5.1. For synthetic data, we use contextual Stochastic Block Models (cSBMs) [2] to generate synthetic graphs with approximately two equal-size classes. For real-world data, we use subgraphs of Cora [28], namely subCora, due to the limitation of computational resources. The details of the generation of synthetic graphs by cSBMs and the generation of subCora graphs are provided in Appendix A.2.1. For the proposed GFL-APPNP, we use a two-layer MLP with 64 hidden units. α is chosen to be 0.1 and the total number of steps for personalized propagation M is set as 10, following the same configuration as it in the [1] for the APPNP model. I is set to be $\{1, 10, 20, 50\}$. SGD is applied as our optimizer with the optimized learning rate. Baseline models including GCN, GAT, and GraphSAGE follow the same design of the well-optimized hyperparameters from [8, 15, 32]. The details for all models are provided in Appendix A.2.2. Table 1 and the first two columns of Figure 3 show that our method of different I can match the performance of the vanilla APPNP on both synthetic graphs and subCora graphs. Our method rivals baseline models based on Table 1.

6.2 Stochastic Node Classification

We also conduct experiments under the SNC setting described in §5.1 to test the robustness of GFL-APPNP. As we know, current graph machine learning models are not designed for the SNC task. Therefore our experiment will focus on the proposed GFL-APPNP. The original cSBMs can not be used to generate data for this task since they are not designed to generate graphs whose nodes have multiple features. We modify the original cSBMs to generate distributions for each client. Details about the modifications for the SNC task are available in Appendix A.2.1. Similar to §6.1, we use the same hyperparameters and the same numbers of updates. The third column of Figure 3 and additional Table 2 provided in Appendix A.2 show that our method is valid for solving the SNC task.

6.3 Supervised Classification

We compare GFL-APPNP, MLPs, and FedMLP under the SC setting described in §5.1. Similar to the SNC task, we modify the original cSBMs to generate distributions for each client. Details about the modifications for the SC task are given in Appendix A.2.1 as well. Both baseline models MLPs and FedMLP share the same model structure, a two-layer MLP model with 64 hidden units. Similar to §6.1, for the proposed GFL-APPNP, we use the same hyperparameters and the same numbers of updates. More details and results are provided in Appendix A.2. The last column of Figure 3 and Table 2 provided in Appendix A.2 shows that our method is valid for solving the SC

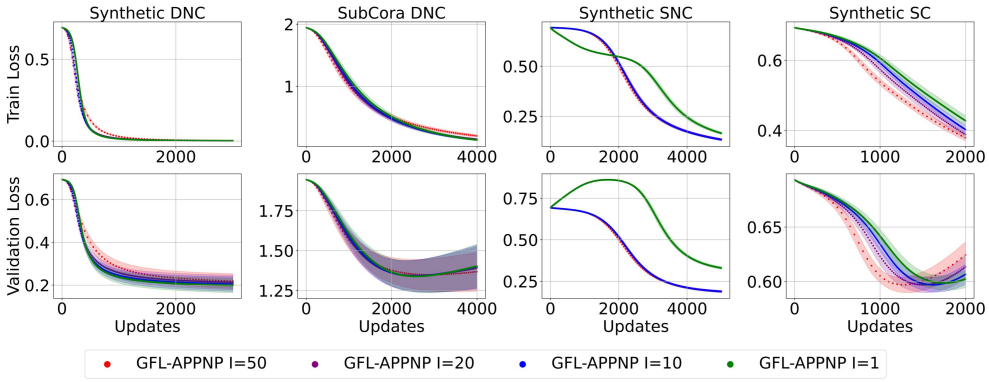


Figure 3: Train and validation loss for DNC (first column), subCora (second column), SNC (third column), and SC (fourth column). For different lines, the numbers of points are different given the same number of updates (if $T = 3000$ and $I = 10$, $3000/10 = 300$ points are in the line). The shaded area represents 95% CIs. See Appendix A.2 for more variants.

task and it demonstrates the necessity of graph in GFL problem as shown in Figure 2.

7 CONCLUSION

In this paper, we formulate Graph Federated Learning on multi-client systems. To tackle the fundamental data sharing conflict between LoG and FL, we propose an FL solution with hidden representation sharing. In theory, we provide a non-convex convergence analysis. Empirically, by experimenting with several classification tasks on graphs, we validate the proposed method on both real-world and synthetic data. Our experimental results show that the proposed method provides an FL solution for GNNs and works for different practical tasks on graphs with a competitive performance that matches our theory.

REFERENCES

[1] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2020. Adaptive universal generalized pagerank graph neural network. *arXiv preprint arXiv:2006.07988* (2020).

[2] Yash Deshpande, Andrea Montanari, Elchanan Mossel, and Subhabrata Sen. 2018. Contextual stochastic block models. *arXiv preprint arXiv:1807.09596* (2018).

[3] Frederik Diehl, Thomas Brunner, Michael Truong Le, and Alois Knoll. 2019. Graph neural networks for modelling traffic participant interaction. In *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 695–701.

[4] Canh T Dinh, Tung T Vu, Nguyen H Tran, Minh N Dao, and Hongyu Zhang. 2021. A New Look and Convergence Rate of Federated Multi-Task Learning with Laplacian Regularization. *arXiv preprint arXiv:2102.07148* (2021).

[5] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948* (2020).

[6] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.

[7] Suyu Ge, Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2020. Graph enhanced representation learning for news recommendation. In *Proceedings of The Web Conference 2020*. 2863–2869.

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[9] Filip Hanzely and Peter Richtárik. 2020. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516* (2020).

[10] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[11] Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Carl Yang, Han Xie, Lichao Sun, Lifang He, Liangwei Yang, Philip S Yu, Yu Rong, et al. 2021. Fedgraphnn: A federated learning system and benchmark for graph neural networks. *arXiv preprint arXiv:2104.07145* (2021).

[12] Chaoyang He, Emir Ceyani, Keshav Balasubramanian, Murali Annavaram, and Salman Avestimehr. 2021. SpreadGNN: Serverless Multi-task Federated Learning for Graph Neural Networks. arXiv:2106.02743 [cs.LG]

[13] Peng Jiang and Gagan Agrawal. 2018. A linear speedup analysis of distributed deep learning with sparse and quantized communication. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 2530–2541.

[14] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*. PMLR, 5132–5143.

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

[16] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997* (2018).

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

[18] Yassine Laguel, Krishna Pillutla, Jerôme Malick, and Zaid Harchaoui. 2021. A superquantile approach to federated learning with heterogeneous devices. In *2021 55th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 1–6.

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

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

[21] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems* 2 (2020), 429–450.

[22] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189* (2019).

[23] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. 2020. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials* 22, 3 (2020), 2031–2063.

[24] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. 2018. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2077–2085.

[25] Yunlong Lu, Xiaohong Huang, Yueyue Dai, Sabita Maharjan, and Yan Zhang. 2019. Blockchain and federated learning for privacy-preserved data sharing in industrial IoT. *IEEE Transactions on Industrial Informatics* 16, 6 (2019), 4177–4186.

[26] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. 2020. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619* (2020).

[27] Chuizheng Meng, Sirisha Rambhatla, and Yan Liu. 2021. Cross-Node Federated Graph Neural Network for Spatio-Temporal Data Modeling. *arXiv preprint arXiv:2106.05223* (2021).

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

[29] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. 2017. Federated multi-task learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 4427–4437.

[30] Sebastian U Stich. 2018. Local SGD converges fast and communicates little. *arXiv preprint arXiv:1805.09767* (2018).

[31] Canh T Dinh, Nguyen Tran, and Tuan Dung Nguyen. 2020. Personalized Federated Learning with Moreau Envelopes. *Advances in Neural Information Processing Systems* 33 (2020).

[32] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[33] Binghui Wang, Ang Li, Hai Li, and Yiran Chen. 2020. Graphfl: A federated learning framework for semi-supervised node classification on graphs. *arXiv preprint arXiv:2012.04187* (2020).

[34] Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. 2019. A semi-supervised graph attentive network for financial fraud detection. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 598–607.

[35] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. 2020. Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 169–178.

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

[37] Han Xie, Jing Ma, Li Xiong, and Carl Yang. 2021. Federated graph classification over non-iid graphs. *Advances in Neural Information Processing Systems* 34 (2021).

[38] Jie Xu, Zhenxing Xu, Peter Walker, and Fei Wang. 2020. Federated Patient Hashing.. In *AAAI*. 6486–6493.

[39] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875* (2017).

[40] Hao Yu, Rong Jin, and Sen Yang. 2019. On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization. In *International Conference on Machine Learning*. PMLR, 7184–7193.

[41] Hao Yu, Sen Yang, and Shenghuo Zhu. 2019. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5693–5700.

[42] Huanding Zhang, Tao Shen, Fei Wu, Mingyang Yin, Hongxia Yang, and Chao Wu. 2021. Federated Graph Learning–A Position Paper. *arXiv preprint arXiv:2105.11099* (2021).

[43] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. 2021. Subgraph federated learning with missing neighbor generation. *Advances in Neural Information Processing Systems* 34 (2021).

A ADDITIONAL WORKS

A.1 Analysis for Gradient Compensation

To solve classification tasks formulated as a GFL problem with hidden representation sharing, as discussed in Section 3.4, we suggest a straightforward gradient estimation strategy, namely gradient compensation. To present the proposed gradient estimation scheme, further define

$$\boldsymbol{h}_{j\to k}^t = \Psi_h(\boldsymbol{x}_j; \boldsymbol{W}_{h,k}^t) \tag{17a}$$

$$\nabla f_k(\boldsymbol{W}_k^t; \Xi_{\mathcal{G}}) = \phi_k(\boldsymbol{h}_{1\to k}^t, ..., \boldsymbol{h}_{N\to k}^t) \tag{17b}$$

where $\boldsymbol{W}_{h,k}^t$ is part of $\boldsymbol{W}_k^t$ for client k at round t and $\boldsymbol{h}_{j\to k}^t$ is the hidden representation of client j for the local updates at client k at round t. Our proposal suggests using the latest aggregated model to compute hidden representations at each communication round. Formally, suppose $t_0 < t$ is the largest multiple of I,

$$\hat{\boldsymbol{h}}_{j\to k}^t = \begin{cases} \Psi_h(\boldsymbol{x}_k; \boldsymbol{W}_{h,k}^t) & j = k \\ \frac{1}{n_j}\sum_{s=1}^{n_j} \Psi_h(\boldsymbol{x}_{j,s}; \bar{\boldsymbol{W}}_h^{t_0}) & j \neq k \end{cases} \tag{18}$$

where $\bar{\boldsymbol{W}}_{h,t_0}$ is part of $\bar{\boldsymbol{W}}_{t_0}$ and n_j is the number of examples drawn according to the distribution of client j. Consequently, the biased gradient estimator employed in local updates at client k has the form of

$$\nabla \hat{f}_k(\boldsymbol{W}_k^t; \xi_k) = \phi_k(\hat{\boldsymbol{h}}_{1\to k}^t, ..., \hat{\boldsymbol{h}}_{N\to k}^t). \tag{19}$$

Our compensation strategy satisfies the guarantee discussed in Assumption 2 with additional assumptions. Specifically, suppose Eq.(16) is used, that is, latest aggregated model is broadcast to estimate the hidden representation, the squared estimation error of $\hat{\boldsymbol{h}}_t^{j\to k}$ is bounded under following assumptions.

Assumption 5. *(Lipschitz Representation Encoder) Hidden representation encoder Ψ_h defined in (3) is Lipschitz continous with constant ρ_h. Formally, $\forall \boldsymbol{W}_h, \boldsymbol{W}_h'$ and $\forall \boldsymbol{x}$, $\exists \rho_{\boldsymbol{x}} > 0$ and $\rho_h := \max_{\boldsymbol{x}} \rho_{\boldsymbol{x}}$ such that*

$$\left\|\Psi_h(\boldsymbol{x}; \boldsymbol{W}_h) - \Psi_h(\boldsymbol{x}; \boldsymbol{W}_h')\right\| \leq \rho_{\boldsymbol{x}} \left\|\boldsymbol{W}_h - \boldsymbol{W}_h'\right\|. \tag{20}$$

Assumption 6. *(Bounded Gradient) Gradient with form ϕ_k has bounded norm. Specifically, for any $k \in [N]$, and $\boldsymbol{h}_1, ..., \boldsymbol{h}_N$, $\exists \Delta_k > 0$ and $\Delta := \sum_{k=1}^N \Delta_k$ such that*

$$\|\phi_k(\boldsymbol{h}_1, ..., \boldsymbol{h}_h)\|^2 \leq \Delta_k. \tag{21}$$

Assumption 5 indicates an uniformly continuity of representation model Ψ_h. Assumption 6 guarantees that unbiased stochastic gradient $\nabla f_k(\boldsymbol{W}_k^t; \Xi_{\mathcal{G}})$ or biased stochastic gradient $\nabla \hat{f}_k(\boldsymbol{W}_k^t; \xi_k)$ is bounded, which ensures $\boldsymbol{g}_t^k$ is bounded and local updates will not make an irreparable deviation. Following Lemma A.1 formally shows that our gradient compensation is valid for the nonconvex result in Section 4,

Lemma A.1. *Consider the federated learning procedure described in Section 3 under Assumptions 5 and 6. Gradient compensation (16) provides estimation with bounded mean squared error. Formally, for any round t, $\forall j \in [N]$ and $\forall k \in [N]$,*

$$\mathbb{E}[\left\|\hat{\boldsymbol{h}}_{j\to k}^t - \boldsymbol{h}_{j\to k}^t\right\|^2] \leq 2\eta^2\rho_h^2 I^2 \Delta_k \tag{22}$$

Lemma A.1 shows that our compensation strategy (16) satisfies that $\sigma_k^2 = 2\eta^2\rho_h^2 I^2\Delta_k$ and $\sigma_H^2 = 2\eta^2\rho_h^2 I^2\Delta$ in Assumption 2. This theoretical guarantee ensures that proposed gradient compensation strategy matches nonconvex results provided in Section 4.

A.2 Supplement to Experiments

A.2.1 Data Generations in Section 6. This part provides a detailed description of the data generation. cSBMs are utilized multiple times, please refer to [2] for a recap of cSBMs.

Synthetic Data for Deterministic Node Classification. In our experiment, 20 synthetic graphs are synthesized, and each of these 20 graphs is generated by using cSBMs with $d = 8, \lambda = 2, \mu = 1, N = 200, p = 100$. Then we fix the graph topology for all 20 generated synthetic graphs. We also fix the train-valid-test split (10%/10%/80%) for all 20 graphs. The training set is balanced and the subgraph induced by the nodes in the training set is connected.

To achieve randomness in repeated simulations, we change the node features for each of these 20 synthetic graphs.

Real Data for Deterministic Node Classification. For real data, we utilized the Cora dataset. Due to our limited computational resources, we extracted subgraphs with 300 nodes from Cora, namely subCora. For each random seed from a pool of random seeds: (1) Randomly select 800 nodes from Cora using one random seed and find all connected components from this subgraph with 800 nodes. (2) For each connected component, we keep it as one training set if it satisfies the following two conditions: (i) Size must fall between 30 and 50 including 30 and 50. (ii) The connected component have all 7 classes, and the standard deviation of the counts for different classes is less than or equal to 2.5. (3) We find all reachable nodes from the training set using BFS with a maximum depth of 2, and select nodes from these reachable nodes as a validation set such that the validation size is equal to the training size. Then select testing nodes so that our subCora graph has 300 nodes in total.

Synthetic Data for Stochastic Node Classification. To generate synthetic data with one label and multiple feature vectors for each node. First, we use cSBMs to generate the graph structure, and we assign a ±1 label to each node by using a Bernoulli distribution with success probability 0.5. The hyperparameters we used to generate the graph structure are $d = 10, \mu = 1, \lambda = 2$ and $N = 200$. Select one training set that is connected and achieves class balance by simply selecting nodes randomly until the subgraph induced by these nodes is connected and it has class balance. Then randomly select the validation set, and testing set such that we have a 10%/10%/80% spilled. Denote the label vector as $\boldsymbol{v} = \{v_1, \ldots, v_N\} \in \{\pm 1\}^N$. And draw a $\boldsymbol{u} \sim N(0, \boldsymbol{I}_p/p)$ where $p = 100$, and it will be used later for feature vectors generation purpose. With these hyperparameters, the ϕ is set to be 0.78. Each node has 40 local data points. The data sampling process for each node is described in the following.

For each node $i = 1$ to $i = N$, (1) the 40 labels for client i is the just the 40 repeats of v_i, denote this vector as $\boldsymbol{y}_i = \{v_i, \ldots, v_i\} \in \{\pm 1\}^{40}$. (2) To generate each of it's 40 local feature vectors denoted as $\boldsymbol{X}_i = \{\boldsymbol{x}_{i,1}^T, \ldots, \boldsymbol{x}_{i,40}^T\} \in \mathbb{R}^{40\times p}$, we utilize the feature generation mechanism from cSBMs, such that $\boldsymbol{x}_{i,j} = \sqrt{\frac{\mu}{N}} v_i \boldsymbol{u} + \frac{\boldsymbol{Z}_{i,j}}{p}$. And just like cSBMs, $\boldsymbol{Z}_{i,j} \in \mathbb{R}^p$ has independent standard normal entries.

We fix the hyperparameters and graph structure, then repeat the sampling procedure for 20 times to generate 20 synthetic graphs that only differ in node/client level feature vectors for random experiments purposes.

Synthetic Data for Supervised Classification. We will use the Bernoulli distribution and multivariate Gaussian distribution to generate labels and feature vectors for each client/node. First, we use the original probabilistic model from cSBMs to generate the graph structure, and we assign a ±1 tag (not labels, which will be generated later.) to each node by using a Bernoulli distribution with success probability 0.5. The hyperparameters we used to generate the graph structure are $d = 5, \mu = 0.1, \lambda = 2.2$ and $N = 50$. Importantly, We make sure that the entire synthetic graph is connected, and this can be easily done by selecting a slightly higher d (average degree) and repeating the graph generation process until the generated synthetic graph is connected. Denote the tag vector as $\boldsymbol{v} = \{v1, \ldots, v_N\} \in \{\pm 1\}^N$. And draw a $\boldsymbol{u} \sim N(0, \boldsymbol{I}_p/p)$ where $p = 100$, and it will be used later for feature vectors generation purpose. With these hyperparameters, the ϕ is 0.96. The data sampling process for each node is described in the following.

For each node $i = 1$ to $i = N$, (1) if the tag for node i (v_i) is -1, assign a Bernoulli distribution with success probability 3/10, else assign a Bernoulli distribution with success probability 7/10. Then draw 120 independent samples from this Bernoulli distribution. These independent samples serve as the local labels for this node i. Denote it as $\boldsymbol{y}_i = \{y_{i,1}, \ldots, y_{i,120}\} \in \{\pm 1\}^{120}$. (2) To generate each of it's 120 local feature vectors denoted as $\boldsymbol{X}_i = \{\boldsymbol{x}_{i,1}^T, \ldots, \boldsymbol{x}_{i,120}^T\} \in \mathbb{R}^{120\times p}$, we utilize the feature generation mechanism from cSBMs, such that $\boldsymbol{x}_{i,j} = \sqrt{\frac{\mu}{N}} y_{i,j} \boldsymbol{u} + \frac{\boldsymbol{Z}_{i,j}}{p}$. And just like cSBMs, $\boldsymbol{Z}_{i,j} \in \mathbb{R}^p$ has independent standard normal entries.

We fix the hyperparameters and graph structure, then repeat the sampling procedure for 20 times to generate 20 synthetic graphs that only differ in clients' data for random experiments purposes. In a supervised classification setting, we utilize data from all clients/nodes and split the local data for each client/node into a train, validation, and test set. In our setting, the train-valid-test split is set to be 8.33%/8.33%/83.33% (i.e., 10/10/100).

A.2.2 Model Description in Section 6. This part provides details about models used in Section 6. For training on all models, including baseline models, we use the SGD optimizer with optimized learning rates based on the specific task and models. During training, we keep tracking the lowest validation loss and the corresponding model and use this model to report test accuracy. For baseline models including `GAT`, `GCN`, and `SAGE`, we do not include layers like dropout on the adjacency matrix layer, no bias. For baseline models, including `FedMLP` and `MLPs` implemented for supervised learning tasks on synthetic graphs, we use a two-layer Multi-Layer-Perceptron (MLP) with 64 hidden units and no bias term.

Graph Federated Learning for APPNP (`GFL-APPNP`). For all experiments we conduct, we use a two-layer MLP with 64 hidden units. We fix the α to be 0.1 and M to be 10, following the `APPNP` model in [16] and [1]. We train `GFL-APPNP` for different $I \in \{1, 10, 20, 50\}$ with gradient compensation and without gradient compensation. Note that different I will lead to different numbers of communications. For example, if we run 3000 updates for $I = 10$, then the number of communications will be $3000/10 = 300$. For the DNC task on both synthetic graphs and subCora graphs, after optimizing the learning rate over $\{0.01, 0.02, 0.05, 0.1, 0.5\}$, we select 0.5 as the best learning rate for synthetic graphs and 0.02 as the best learning rate for subCora. We run 4000 updates for subCora graphs and 3000 updates for synthetic graphs. The same learning rates and the numbers of updates are adapted for both variants with gradient compensation and variants with no compensation. For the SNC task on synthetic graphs, we use 0.2 as learning rate and batch size of 40 (full batch) and run 5000 updates for each variant with gradient compensation. For SC tasks, we use a learning rate of 0.2 and batch size of 5 and run 2000 updates for each variant with gradient compensation.

Baseline Models. In our experiments, baseline models include `GCN` [15], `GAT` [32], `GraphSAGE` [8] for DNC setting and Fedederated Learning for Multi-Layer-Perceptron (`FedMLP`) and `MLPs` for SC

Table 2: Summary table for average test accuracy and 95% confidence interval on all experiments and all models considered. Hyphens indicate a specific model is not applicable under certain task, and where V1 represents noisy gradient compensation, V2 represents no gradient compensation. We denote deterministic node classification as DNC, stochastic node classification as SNC, and supervised classification as SC.

	Synthetic DNC	SubCora DNC	Synthetic SNC	Synthetic SC
GFL-APPNP $I = 1$	93.2 ± 0.92%	54.2 ± 3.69%	98.7 ± 0.26%	70.0 ± 0.32%
GFL-APPNP $I = 10$	93.4 ± 0.99%	54.1 ± 3.72%	92.4 ± 0.19%	70.0 ± 0.36%
GFL-APPNP $I = 20$	93.3 ± 0.94%	54.3 ± 3.73%	92.5 ± 0.17%	70.0 ± 0.30%
GFL-APPNP $I = 50$	93.0 ± 0.96%	54.0 ± 3.73%	92.5 ± 0.17%	70.2 ± 0.33%
GFL-APPNP-V1 $I = 10$	93.1 ± 0.83%	54.0 ± 4.12%	99.2 ± 0.26%	68.7 ± 0.42%
GFL-APPNP-V2 $I = 10$	82.3 ± 2.09%	47.3 ± 3.81%	90.7 ± 0.28%	69.0 ± 0.45%
APPNP	93.2 ± 0.92%	54.2 ± 3.69%	—	—
GCN	95.2 ± 0.54%	51.9 ± 3.78%	—	—
GAT	93.3 ± 1.03%	47.9 ± 3.01%	—	—
GraphSAGE	70.2 ± 4.21%	47.0 ± 3.73%	—	—
FedMLP $I = 10$	—	—	—	61.0 ± 0.54%
FedMLP $I = 20$	—	—	—	61.0 ± 0.46%
FedMLP $I = 50$	—	—	—	70.0 ± 0.32%
MLPs	—	—	—	61.0 ± 0.60%

setting. We use GCN with two layers with 64 hidden units following [15] and learning rate 0.1 for synthtic data and 0.01 for subCora. For GAT, we use 2 layers where the first layer has 8 attention heads and 8 hidden units per head, the second layer has 1 attention head and 64 hidden units following [32] and learning rate 0.02 for synthtic data and 0.01 for subCora. Both GCN and GAT are trained by 4000 updates. In GraphSAGE, we use 2 layers with 64 hidden units. For the task on synthetic graphs, we train 5000 updates with a learning rate equal to 0.02. For task on subCora, we train 4000 updates with a learning rate equal to 0.01. For the SC task, we train FedMLP (different $I \in \{10, 20, 50\}$) with 2000 updates paired with a learning rate of 0.1 and a batch size of 5 while we train independent MLPs on clients 200 updates with a learning rate of 0.1 and a batch size of 5.

A.2.3 Additional Experiment I: Effect of Graph Connectivity (Figure 2). In this section, we provide details for Figure 2(a). We use the same data generation process in A.2.1 for supervised classification to generate four synthetic graphs with different connectivity measured by $\lambda_{\max}(\boldsymbol{B}_N \boldsymbol{L}^\dagger)$, each synthetic graph will repeat the data sampling process 20 times to achieve randomness. We use the same model which is our method with 64 hidden units, $I = 10$, and same model initialization to conduct experiments on all four synthetic graphs and their corresponding 20 repetitions. All four synthetic graphs has the same hyperparameters $N = 40, \mu = 1, \lambda = 2,$ and $p = 100$ expect for one hyperparmeter $d \in \{25, 15, 10, 5\}$, thus they all have the same $\phi = 0.574$. Recall that d represents the average degree for a synthetic graph, so a higher d naturally leads to a higher connectivity, and the results are four different $\lambda_{\max}(\boldsymbol{B}_N \boldsymbol{L}^\dagger)$ values $\{1.76 \times 10^{-3}, 4.17 \times 10^{-3}, 1.09 \times 10^{-2}, 7.52 \times 10^{-2}\}$. All nodes have the same number of local data points 120 for all four synthetic graphs, and the train-valid-test split is 10/10/100. SGD optimizer is used to train 1500 updates for our method with a learning rate 0.5, and a batch size of 5 for all four synthetic graphs.

A.2.4 Additional Experiment II: Necessity of Graph Structure in GFL (Figure 2). As our discussion in Section 3, the network of clients in multi-client systems accounts for the statistical heterogeneity problem. This empirical study aims to show that the heterogeneity problem is non-negligible. Our experiment is under SC setting with baseline models MLPs and FedMLP, matching the same setting in Section 6.3. The result is given by boxplot on Figure 2(b). The models and details are the same as Section 6.3.

A.2.5 Additional Experiment III: Noisy gradient compensation. Instead of uploading local gradient $\nabla \boldsymbol{h}_k^t$ of each client k at each communication round t where $t \mod I = 0$ to central server, one can add a random noise vector denoted as $\boldsymbol{\epsilon}_k^t$ that has independent standard normal entries to the gradient vector, and upload $\nabla \boldsymbol{h}_k^t + \boldsymbol{\epsilon}_k^t$ to central server for better node-level privacy. We provide additional experiments that include noisy gradient compensation strategy with $I = 10$ with the same data and model from previously mentioned experiments. Also, we use the SGD optimizer for noisy gradient and hyperparameters including learning rate, batch size, and the number of updates are the same as the case without Gaussian noise.

A.2.6 Summary Table and Figure for All Experiments. A test summary Table 2 for all experiments and models is provided in this section. The experimental results using our method of $I = 10$ **without gradient compensation** (i.e. only upload h_k^t to the central server for each client k) are also added to the summary table and figure for reference. For DNC, and SC tasks, we use the same number of updates, learning rate, and batch size in A.2.2 for our method without gradient compensation. For the SNC we change the learning rate from 0.2 to 0.6 and the number of updates from 5000 to 8000 for our method without gradient compensation.