Dynamic Activation of Clients and Parameters for Federated Learning over Heterogeneous Graphs

Zishan Gu
Columbia University
New York, USA
zg2409@columbia.edu

Ke Zhang
University of Hong Kong
Hong Kong, China
cszhangk@connect.hku.hk

Guangji Bai
Emory University
Atlanta, USA
guangji.bai@emory.edu

Liang Chen
Sun Yat-sen University
Guangzhou, China
chenliang6@mail.sysu.edu.cn

Liang Zhao
Emory University
Atlanta, USA
liang.zhao@emory.edu

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

ABSTRACT

The data generated in many real-world applications can be modeled as heterogeneous graphs of multi-typed entities (nodes) and relations (links). Nowadays, such data are commonly generated and stored by distributed clients, making direct, centralized model training unpractical. While the data in each client are prone to local bias distributions, generalizable global models are still in frequent need for large-scale applications. However, the large number of clients enforce significant computational overhead due to the communication and synchronization among the clients, whereas the biased local data distributions indicate that not all clients and parameters should be computed and updated at all times. Motivated by specifically designed preliminary studies on training a state-of-the-art heterogeneous graph neural network (HGN) with the vanilla FedAvg framework, in this work, we propose to leverage the characteristics of heterogeneous graphs by designing dynamic activation strategies for the clients and parameters during the federated training of HGN, named FedDA. The effectiveness and efficiency of our proposed techniques are backed by both theoretical and empirical analysis– We mathematically illustrate its efficiency gain; meanwhile, we demonstrate the significant performance gains of FedDA and corroborate its efficiency gains with extensive experiments over multiple realistic FL settings synthesized based on real-world heterogeneous graphs.

KEYWORDS

heterogeneous graphs, federated learning, dynamic activation, performance gain, efficiency gain

*Corresponding author.

FedGraph '22, October 21, 2022, Atlanta, GA, USA

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

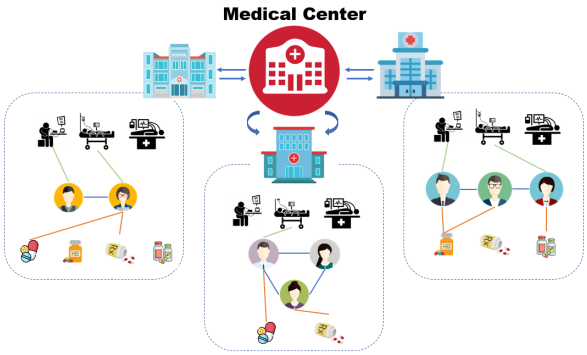


Figure 1: Toy example of a real-world clinical heterograph.

1 INTRODUCTION

Heterogeneous graphs (heterographs) constructed with multiple types of nodes and links can well record and model complex real-world application scenarios [40], such as citation prediction on bibliographical networks [29], patient profiling on healthcare networks [30]), emergency medical service [37] and recommender systems [4, 12, 19, 33]. Herein, we take the healthcare system as an example, as illustrated in Fig. 1. For each hospital, it records the patients' healthcare profiles independently, which contain various types of information, such as demographics, laboratory testing results, medical treatments, and diagnosis histories. By regarding patients, drugs, procedures, and diseases as nodes of different types and linking every patient with his/her prescribed drugs, operated procedures, diagnosed diseases, and other closely interacted patients, the hospital possesses a clinical heterograph.

Within one domain, there can be numbers of service providers (*e.g.*, clinics). In contrast to general centralized global data, for each data owner, the locally collected and preserved data can be prone to selection biases. In the healthcare system example, a clinic with the specialty in heart diseases can attract more patients with the corresponding medical needs, whereas a clinic possessing a prestigious psychology department can collect more data regarding psychological disorders. Meanwhile, the heart disease clinic can record more links between surgeries and patients than the psychological disease clinic. Thus, each data owner possesses a biased local subset of the global clinical data. When each of them constructs a local clinical heterograph, it can be regarded as a non-independent-identically-distributed (Non-IID) heterogeneous subgraph of the

entire global heterograph containing all potential healthcare data generated such as in the city. Due to privacy protection regulations and interest conflicts, clinics cannot directly share their Non-IID heterographs with others. However, given a city-level task such as pandemic prediction, the question arises as to how to efficiently obtain a generalized global heterograph model without actually putting all Non-IID heterographs of different data owners together.

Federated learning (FL) [17] was proposed to enable distributed data owners (clients) to collaboratively learn a global model without sharing local raw data. Originally designed for traditional machine learning tasks such as in Computer Vision (CV) and Natural Language Processing (NLP), FL exhibits enormous potential in effectiveness, especially when data are IID across data owners [48]. Recently, FL also achieves remarkable progress in tackling relational data mining tasks, *i.e.*, for graph learning [3, 5, 9, 39, 43]. However, they primarily focus on homogeneous graphs (homographs), which cannot handle the diversity of node and link types.

Key Findings As the first work on FL over distributed Non-IID heterographs, we first conduct a preliminary analysis to examine the pivotal factors that determine the utility of the outcome graph learning model. Specifically, we test the state-of-the-art heterograph model of Simple-HGN [20] with the FedAvg protocol [22] across our distributed system. Our empirical results surprisingly and clearly show that there is a potential for attaining a global model with better performance yet smaller cost not only with fewer clients involved in each round of FedAvg but also by gathering only partial gradients from involved clients. Notwithstanding, when we randomly reduce the transmitted data along the FL process, the global model can exhibit more unstable performances. Based on the observations in our preliminary analysis, we further formulate two natural yet unique challenges for FL over distributed heterographs.

Challenge: *How to dynamically activate clients and parameters to obtain the final generalized global model efficiently?*

Solution: FedDA. Utilizing the returned information from each clients, we design an FL process termed FedDA. Technically, FedDA dynamically selects clients according to their returned gradients in the previous round. In particular, clients returning gradients smaller than the averaged values will be deactivated in the next round. We theoretically justify the effectiveness of our client and parameter selection approach based on previous gradients. Note that, although FedDA is motivated by our distributed heterogeneous graph setting, it can potentially generalize to other types of data in the Non-IID FL setting.

Approaches

We provide two strategies to implement the dynamic activation of clients and parameters in FedDA towards improved performance and efficiency. To empirically verify the utility of our proposed methods, we perform experiments on five distributed heterograph systems realistically synthesized with different numbers of clients over two real-world benchmark datasets of heterographs from different application scenarios . Experimental results indicate that FedDA can lead to a global model with significantly improved performance and efficiency In-depth training analysis and hyperparameter studies further corroborate the improved convergence, stability, and robustness of FedDA.

2 RELATED WORKS

2.1 Federated learning on graphs

Federated learning (FL) [18, 42] on graphs has drawn significant attention from researchers in relevant fields. Existing works on FL over graph data mainly consider homogeneous graphs (homographs), whose nodes and links only have a single type. In the vertical FL setting, [23, 46] studied the scenarios where node features and structures on distributed graphs vary across local devices. In the more common horizontal FL setting, [9] proposed an open-source benchmark system for federated GNNs; [39] proposed a clustered FL framework for the graph-level task across graphs from different domains; [3, 5, 43] studied the distributed graph setting under the consideration of cross-graph links, but they have only studied it on homographs.

With the consideration of multiple types of links, [38] studied FL over the bipartite user-item graph for recommender systems, and [6] studied FL over knowledge graphs for their completion. In complex real-world applications, heterographs that include multiple types of both nodes and links can better simulate and model realistic networks compared to knowledge graphs [29, 40]. The superior richness of information in heterographs not only creates an emerging need of applying FL over them, but also brings in unique challenges, such as the Non-IIDness over different types of data.

To the best of our knowledge, this is the first work towards effective FL over distributed Non-IID heterographs. Besides improving the performance compared to basic FL, our proposed approach further steps out to uplift the communication efficiency.

2.2 Strategic training of GNNs

Similar to traditional machine learning models, when the testing heterograph is considered as a generic graph following the global data distribution, training a GNN on biased (Non-IID) local heterographs degenerates its test performance [16, 44]. Several training strategies designed for CV and NLP [14, 27, 28, 35] tasks in solving the Non-IID problem have been transferred to graph learning tasks.

For example, inspired by cognitive science, curriculum learning [1] points out the pivotal role of training sample selection in machine learning tasks, which is now applied in graphs as well. [36] proposed a novel methodology for curriculum learning on graphs that includes novel graph evaluation and selection steps. Nevertheless, it tackles the graph-level task (graph classification) and does not consider the node-level and link-level tasks inside every graph. Another aspect of tackling the label imbalance issue in the Non-IID graph learning task is to leverage negative sampling skills [24]. [11] focused on the recommendation task with knowledge graphs and proposed a method to generate negative samples by leveraging a hop mixing technique on selected neighbors.

However, all these works are restricted to the locally (centralized) training setting instead of the distributed setting. Existing distributed GNN algorithms can be classified into two types: "*partition-based*" methods [13, 26] partition the graph into different subgraphs by dropping the edges across subgraphs, thus enjoying low communication cost but suffering from information loss. On the other hand, "*propagation-based*" methods [21, 31, 34, 45, 47] do not ignore the edges across different subgraphs with neighbor communications among subgraphs to satisfy GNN's neighbor aggregation but

may suffer from communication overhead. In summary, our work considers the FL scenario over distributed graphs, which can jointly handle the distributed setting and the privacy issue while the existing works above cannot.

3 PROBLEM FORMULATION

System We denote a global heterograph as $H = \{V, E, \varphi, \psi, X\}$. V includes all nodes in H, where each node $v \in V$ is associated with a node type $\varphi(v)$ and attributed with a feature vector $x_v \in X$ with dimension $d_{\varphi(v)}$. E denotes the set of all links in H. Each $e \in E$ is associated with an edge type $\psi(e)$, which is determined by the types of nodes on its two ends. Note that in this work, we consider heterographs without multiple types of links between two specific types of nodes, but our methods extend trivially beyond this constraint.

In the FL setting, we have a central server S, and M clients $\mathcal{D} = \{D_i | i \in [M]\}$ with distributed sub-heterographs $\mathcal{H} = \{H_i | i \in [M]\}$ (here $[M]$ denotes the set of integer from 1 to M). Slightly different from H, we denote $H_i = \{V_i, E_i, \varphi, \psi, X_i\}$ as the sub-heterograph of H owned by D_i, for $i \in [M]$. While the global graph H conceptually exists, no entity is able to aggregate all sub-heterographs to really obtain H.

In a distributed heterograph system, it is common to see the non-identical distribution of edge types across local devices as data are generated from heterogeneous applications or locations. For example, in community clinics, diagnosing patients with lymphoma and operating craniotomy on patients are much rarer than those in national hospitals. Similarly, for an online music application, songs in a certain language (*e.g.*, Japanese) are far more likely to be favored by users from certain locations (Asia-Pacific).

In this paper, we mainly focus on the issue of local Non-IID edge types and formulate the problem and respective assumptions as follows.

Problem According to the system described above, the global graph H has its all nodes and edges distributed in M sub-heterographs. Specifically, we have $V = V_1 \cup \cdots \cup V_M$, and $E = E_1 \cup \cdots \cup E_M$. For $i, j \in [M]$ and $i \neq j$, the system allows overlaps, *i.e.*, $|V_i \cap V_j| \geq 0$ and $|E_i \cap E_j| \geq 0$.

We denote the edge type distribution for an edge set E_i as the probability $\mathcal{P}_i = P(\psi(e) | e \in E_i))$. For any two different clients D_i and D_j, we have $\mathcal{P}_i \nsim \mathcal{P}_j$, which we refer to as biased data regarding Non-IID link types in distributed heterographs that we consider in the FL setting.

We consider the downstream task of link prediction over node pairs in H, which is one of the most common tasks on heterographs [20]. Technically, for a pair of nodes on the heterograph, link prediction infers whether there exits an edge between them. Therefore, we formulate our goal of federated link prediction on heterographs as follows.

Goal The system exploits an FL framework to collaboratively learn on isolated sub-heterographs, $\{H_i\}_{i \in [M]}$, across M clients, without direct data sharing, to obtain a global link prediction model F with parameters indexes $\mathcal{I}$. We formulate the link prediction task as a binary classification problem. The N learnable parameters θ in F are optimized on queried pairs of nodes and their neighbors that are similar to the ones drawn from the global heterograph H, and finally test F on the global test data with all types of edges. We formulate the problem as finding θ^* that minimizes the loss $\mathcal{L}$ on H by aggregating local loss values as

$$\theta^* = \arg\min \mathcal{L}\big(F(\theta|H)\big) = Agg\Big(\mathcal{L}_i\big(F_i(\theta|H_i)\big)\Big), \tag{1}$$

where $Agg(\cdot)$ is an aggregation function (*e.g.*, averaging). $\mathcal{L}_i$ is the local empirical loss defined as

$$\mathcal{L}_i\big(F_i(\theta|H_i)\big) \coloneqq \mathbb{E}_{\psi(e)\sim\mathcal{P}_i}\left[\ell\left(F_i\left(\theta; H_i(v), H_i(u)\right), \psi(e)\right)\right],$$

where e is the link between u and v, $H_i(v)$ is v's ego-graph (v and its neighbors) on H_i, and ℓ is a task-specific loss function such as cross-entropy for link classification.

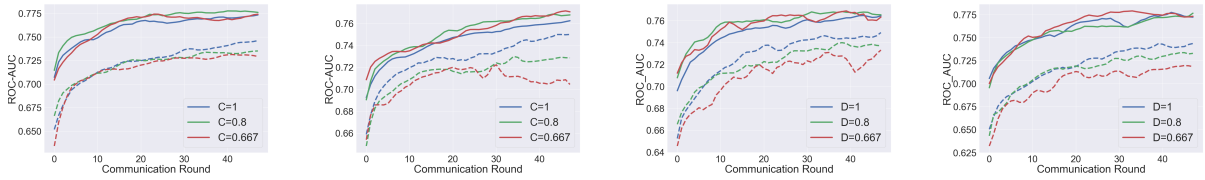
Under basic FL frameworks (*e.g.*, FedAvg), learning from Non-IID data can rapidly distort the outcome model's performance compared to the one trained on IID data (drop at most 48.3% in certain CV tasks [10]). For the complex and highly irregular heterographs, it is an urgent demand to design an effective and efficient FL method for the Non-IID setting. Herein, we conduct preliminary studies on Non-IID heterographs to find out pivotal factors in FedAvg that determine the outcome graph learning model's utility.

4 MOTIVATING PRELIMINARY STUDIES

Here, we present intuitive preliminary studies to investigate the problem of heterograph FL across the distributed system. Specifically, we adopt the widely used weights-sharing-based FL protocol, FedAvg [22], with the state-of-the-art heterograph model of Simple-HGN [20] on a small subgraph from the benchmark heterogeneous network of DBLP [41]. We are here to observe how Simple-HGN performs under the vanilla FedAvg framework with biased and unbiased data in the global downstream task of link prediction. We construct the following experiments trying to answer these questions: Should we activate all clients when data are largely biased? Should we aggregate all parameters when the data are largely biased? What are some potential enhancements on the traditional FedAvg framework when applied on the distributed heterographs?

Simple-HGN [20] studied 12 recent state-of-the-art heterogeneous graph neural networks (HGNs) and found that vanilla Graph Attention Networks (GAT)[32], one of the homogeneous GNNs, can surprisingly outperform all existing HGNs. They further proposed a simple but strong baseline by improving GAT with adaptions to heterograph called simple-HGN, which is shown to be significantly better than all previous models. Specifically, simple-HGN follows the common encoder-decoder model structure to perform link prediction. In the encoder part, they extended the original graph attention mechanism by including edge type information into the attention calculation utilizing different edge embeddings. Moreover, they also extended the network structure with residual connections between layers and L2 normalization in the final output. As for the decoder module, they incorporate dot product and DistMult as the score function, which are proved to be suitable for different datasets. For more details of Simple-HGN, please refer to Sec.5.1.1 and their original paper [20].

Federated Averaging (FedAvg) [22] proposed FedAvg as a FL framework to jointly train a global model across distributed clients, under which a server will activate clients and distribute the global model at the beginning of each round, and each client will train a



(a) (b) (c) (d)

Figure 2: Performance curves of FedAvg with different client activation rate (C) and parameter activation rate (D). Fig.2(a) and 2(c) are the results on unbiased data (IID link types) across clients, whereas Fig.2(b) and 2(d) are the results on biased data (Non-IID link types) as discussed in Section 3. The results we show are max and min scores over five separate runs, in which the solid lines are the best performance in each round, while the dotted lines are the worst.

model locally with the same objective function and learning rate. The resulting global model's weights are updated by the average of all returned gradients. They also pointed out that the same initialization for local training can produce significant loss reduction during training, thus it is crucial for this framework that clients train their model with the uniformly distributed global weights starting from the very first round.

4.1 Motivating Observations

We run Simple-HGN under the vanilla framework of FedAvg and display the trends of model performances along communication rounds. In Fig.2(a) and Fig.2(b), we randomly select gradients from C-fraction of clients to aggregate in each round, while in Fig.2(c) and Fig.2(d), we randomly select gradients of D-fraction of parameters to aggregate. We come up with the following two observations according to the results of our preliminary studies.

Observation 1: With distributed heterographs, more clients involved in each round does not ensure better performance. As the results in Fig.2(a) and Fig.2(b) indicate, when we consider the best performed models over five runs (solid lines), the 80% and 67% clients versions both achieve comparable or even better performance than the 100% clients version. However, as for the worst models (dashed lines), activating less clients may end up with much worse models, especially when edges types are Non-IID. That is to say, decreasing the number of involved clients does not harm the optimal performance yet introduces more instability, which implies that we may be able to simultaneously enhance the model performance and communication efficiency by activating less clients as long as we can find the right group to activate.

Observation 2: When we randomly update only a part of the parameters in FedAvg, we can still obtain a well-performed global model. Here we compare FedAvg with its partially aggregating variations. Specifically, during the aggregation process, instead of taking the weighted average of all gradients as in FedAvg, we only take the average of a randomly selected set of gradients with ratio D. As shown in Fig.2(c) and Fig.2(d), similarly to activating part of the clients, activating less parameters may also end up with models just as powerful (solid lines), while this random selection also results in worse performance in the worst case (dashed lines). Nevertheless, this also implies potential enhancement over FedAvg by reducing the number of returned gradients. Specifically, if we deliberately choose the set of returned gradients for each client, we may be able to end up with a model just as powerful with less communication cost.

Summary Observation 1 indicates that we can potentially achieve the same or even better performance with less clients involved for each round, while Observation 2 implies that we may be able to gather only a part of the gradients from each clients without harming the final global model performance. However, according to the experimental results, there is a need to design appropriate strategies to adaptively choose groups of clients to contribute as well as the gradients we want to aggregate along the FL process. In an ideal case, through leveraging smaller sets of clients and gradients in each round, such strategies can lead to great enhancement in communication and computation efficiency, which is an essential merit in the distributed graph learning system. In the meantime, these strategies can potentially obtain global models with similar or even better performance.

Can we design a proper clients and parameter selecting strategy for FL to reduce the instability, benefit the data transmission efficiency, while assist the system in achieving a satisfactory result simultaneously? We leave the discussion of our novel FL framework designs in the following section.

5 METHODOLOGY

As we discussed before, in our FL setting, sub-heterographs are independently collected with potential selection bias. Based on the observations of our preliminary studies in Section 4.1, we design a FL framework with dynamic activation of clients and parameters (FedDA) on top of FedAvg to alleviate the negative effect brought by local bias and elevate the communication efficiency simultaneously. The overall structure of FedDA is presented in Fig.3.

In this section, we first discuss the technical details of Simple-HGN and FedAvg. Then we present the specific methods to dynamically activate clients and parameters during the training of FL. Finally, we present our proposed FedDA.

5.1 Simple-HGN and FedAvg

5.1.1 Simple-HGN. As described in Sec.4, Simple-HGN (S-HGN) is adopted from GAT with three enhancements: Learnable type-wise edge embedding, residual connections between layers, and L_2 normalization on the final output.

Learnable Edge-type Embedding At each layer l, S-HGN adopts a d_l-dimensional embedding vector $\vec{r}^{\,l}_{\psi(e)}$ in order to include the

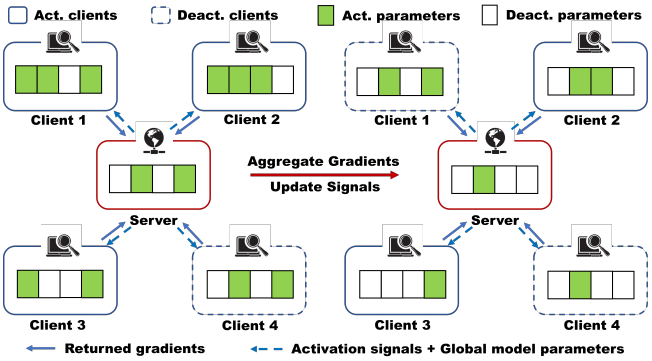


Figure 3: Illustrative visualization of FedDA. Best viewed in color. The clients in dotted boxes and the parameters in white are the deactivated ones. For each round, the server sends the activation signals and global model to each activated client. The activated clients update the model locally for several epochs and return the gradients of the activated parameters accordingly. Finally, the server gathers the returned gradients and update the global model as well as the activation signals.

edge type information into the graph attention mechanism, where $\psi(e)$ is the type of edge e. The attention score over edge e between node u and node v is calculated as following[1]:

$$\hat{\alpha}_{uv} = \frac{\exp\left(\sigma\left(\vec{a}^T\left[W\vec{h}_u||W\vec{h}_v||W_r\vec{r}_{\psi(e)}\right]\right)\right)}{\sum_{k\in\mathcal{N}_u}\exp\left(\sigma\left(\vec{a}^T\left[W\vec{h}_u||W\vec{h}_k||W_r\vec{r}_{\psi(e)}\right]\right)\right)}, \tag{2}$$

where W_r is a learnable matrix to transform type embeddings, and σ is LeakyReLU as the activation function.

Residual Connections S-HGN adopts pre-activation residual connection for node representations across layers for link prediction tasks. The aggregation function at layer l is constructed as

$$\vec{h}_v^{(l)} = \sigma\Big(\sum\nolimits_{v\in\mathcal{N}_u}\alpha_{uv}^{(l)}W^{(l)}\vec{h}_v^{(l-1)} + W_{res}^{(l)}\vec{h}_u^{(l-1)}\Big), \tag{3}$$

where $\alpha_{uv}^{(l)}$ is the attention weight over edge e between node u and node v , and the activation function here is ELU [7] by default.

L_2 **Normalization** S-HGN performs L_2 normalization on the final output embeddings before presenting it to the decoder, which makes the dot product of embeddings being equivalent to their cosine similarity. The normalized embedding for node v is calculated as $o_v = \vec{h}_v^{(L)}/||\vec{h}_v^{(L)}||$, where L is the total number of layers.

Incorporating with these three enhancements, accompanied with the multi-head attention mechanism of vanilla GAT, S-HGN is claimed to achieve the best performance on hetro-graphs with link prediction task.

5.1.2 FedAvg. FedAvg[22] is a FL framework for training deep neural networks across decentralized data from clients based on iterative averaging. Specifically, in each round t, the server first broadcasts the latest global model's parameters $\mathbf{w}^{(t)}$ to all activated clients. Then, for each client i, it lets $\mathbf{w}_i^{(t)} = \mathbf{w}^{(t)}$ and then updates the received global model locally as following:

$$\mathbf{w}_i^{(t+1)} \leftarrow \mathbf{w}_i^{(t)} - \eta_i \cdot \nabla\mathcal{L}_i(\mathbf{w}_i^{(t)}, \mathcal{B}_i), \quad \forall\, i \in [M] \tag{4}$$

where η_i is the learning rate for local update on client i and $\mathcal{B}_i$ is the mini-batch sampled from local data on client i. Finally, the server aggregates the local models and takes their weighted average as the updated global model by round t. However, due to privacy concerns, we assume the server doesn't own prior knowledge of the distribution of global and local data, thus we perform the average without discriminative weights as

$$\mathbf{w}^{(t+1)} \leftarrow \sum\nolimits_{i=1}^{M} p_i\mathbf{w}_i^{(t+1)} \tag{5}$$

where p_i is the aggregation weight and for standard FedAvg it follows $p_i = \frac{1}{M}, \forall\, i$.

5.2 Dynamic Activation of Clients

From our *Observation 1* in Section 4.1, the total number of transmitted gradients can be potentially reduced by activating only a portion of the clients. Thus, to control the client selection, the server maintains an activation set $\mathcal{D}_A^t$ containing the clients to be activated in round t. Here, we present a dynamic activation mechanism for clients to deliberately choose $\mathcal{D}_A^t$ during the training process of FedDA.

Intuitively, we want to leave out clients with rather trivial contributions in each round. In other words, if most of the returned gradients from a client are smaller than the average of all clients, we deactivate this client in the next round. However, we cannot obtain the contribution of a client in each round without actually gathering it. To deal with this problem, we estimate the more effective set of clients in the next round using the returned gradients from last round.

Furthermore, if we continuously deactivate clients and update $\mathcal{D}_A$ over training, the number of involved clients may be reduced rapidly and the training will be much biased. Thus, to guarantee enough exploration and exploitation of clients during FedDA and prevent the global model from falling into some biased local optimum, we propose two different strategies as following.

Restarting-based Herein we introduce a hyper-parameter $\beta_r \in (0, 1)$ as a threshold value controlling the number of activated clients. Particularly, when $|\mathcal{D}_A^{(t+1)}| < \beta_r M$, $\mathcal{D}_A^{(t+1)}$ is set to the initial values, *i.e.*, $\mathcal{D}_A^{(t+1)} \leftarrow \mathcal{D}$ In this way, after the restart process, the server gets all clients back involved with all their latest parameters. Therefore, it can retrieve a global monitoring of the capability of each client regarding the current state of the global model so as to make a more informed decision in the following communication rounds.

Exploration-based The main idea of the exploration-based strategy is to maintain a certain size of the $\mathcal{D}_A^{(t)}$ with a hyper-parameter $\beta_e \in (0, 1)$. Technically, when $|\mathcal{D}_A^{(t)}| < \beta_e M$, the server randomly explores $(\beta_e M - |\mathcal{D}_A^{(t)}|)$ clients from the deactivated clients, *i.e.*, $\mathcal{D} \setminus \mathcal{D}_A^{(t)}$, to maintain the number of activated clients as $\beta_e M$. With this approach, the server can exploit more diverse training information from the system and alleviate the selection bias or errors in choosing the activated clients. Note that, to make this process

[1]We omit the superscript l in the equation for brevity.

more historically consistent, we do not consider the clients that has just been deactivated in this round to rejoin $\mathcal{D}_A$ in the next round.

5.3 Dynamic Activation of Parameters

From our *Observation 2* in Section 4.1, the number of transmitted gradients of a single activated client can also be potentially reduced since we can achieve a global model just as powerful by aggregating only a part of the gradients. To achieve this, the server preserves a request indices set $\mathcal{I}^{(t)} = \{\mathcal{I}_i^{(t)} | i \in \mathcal{D}_A\}$, where $\mathcal{I}_i^{(t)} = \{0,1\}^N$ indicates the required parameter indices for a client i in round t. For $k \in [N]$, $\mathcal{I}_i^{(t)}[k] = 1$ means the server is requesting $\theta_i^{(t)}[k]$ from D_i in round t, and otherwise if $\mathcal{I}_i^{(t)}[k] = 0$.

In fact, the contribution of a client to certain sets of the parameters can be trivial due to the local training data bias. For example, in the most extreme case, if the downstream task on a client only perform link prediction on one single type of links, then the weights in the decoder of S-HGN for other type of links will not be properly trained, which means aggregating them all is equivalent to giving the parameters in the last round a residual weight starting from the very beginning with the random initialization. In this spirit, we design the central server to first consider the distribution of returned gradients and come up with a threshold for each parameter. Then, it signals the clients to only return the gradients of parameters that are above that threshold[2].

Similar to the deactivation of clients, we approximate the larger set of gradients in the next round using the returned gradients from last round. That is, for the returned gradient $g_{(i,k)}^{(t)}$ of parameter k from client i at round t, if it is smaller than the average value of all the gradients for this parameter at round t, we ask i not to pass the gradient of g in round $t+1$. Furthermore, with partially activated parameters, we can extend the deactivation of clients discussed in Sec.5.2 by considering the amount of contributed gradients. Thus, we introduce a hyper-parameter α to control the occupation rate of the client, *i.e.*, whether a client is sufficiently contributing to the FL training process. For $i \in [M]$, if the portion of the returned gradients compared with the total number of disentangled parameters N_d is less than α, we give up this client in the next round. That is, if $\sum_{k\in[N_d]} \mathcal{I}_i^{(t)} < \alpha \cdot N_d$, $\mathcal{D}_A^{(t+1)} = \mathcal{D}_A^{(t)} \setminus \{D_i\}$.

5.4 The FedDA Algorithm

In this section, we introduce the specific algorithms of FedDA, of which the overall pipeline is shown in Algorithm 1.

5.4.1 Deactivation of Clients and Parameters. The initialization for $\mathcal{D}_A^{(0)}$ is $\mathcal{D}$, and $\forall\, D_i \in \mathcal{D}_A^{(0)}$, $\mathcal{I}_i^{(0)}$ is set to ones.

For the t-th round of FedDA, the server first broadcasts the current model weights $\theta^{(t)} = \{\theta^{(t)}[k] | k \in [N]\}$ and $\mathcal{I}^t$ to the clients in $\mathcal{D}_A$, and then each activated client locally updates the received model to $\hat{\theta}_i^{(t)} = \{\theta_i^{(t)}[k] | k \in [N]\}$. After collecting weights from the activated clients, for $k \in [N]$, the server updates each $\theta^{(t)}[k]$ as

$$\theta^{(t+1)}[k] = \frac{1}{\sum_{D_i \in \mathcal{D}_A^{(t)}} I_i^{(t)}[k]} \sum \left\{ \theta_i^{(t)}[k] | \mathcal{I}_i^{(t)}[k] = 1 \right\}. \quad (6)$$

[2]In this work, we set that threshold to be the mean value, and leave the discussion of other settings to future work.

Algorithm 1 *FedDA.*

Require: Number of total communication rounds T, local batch size B, number of local epochs E; indices of all the trainable parameters $[N]$, indices of all the disentangled parameters $[N_d]$.

Server executes:
  initialize θ
  for each round $t \in [T]$ **do**
    for client $i \in D_A^{(t)}$ **do**
      $\theta_i^{(t)} \leftarrow \text{ClientUpdate}(i, I_i^{(t)}, \theta^{(t)})$
    end for
    for parameter $k \in [N]$ **do**
      $\theta^{(t+1)}[k] = \frac{1}{\sum_{D_i \in \mathcal{D}_A^{(t)}} I_i^{(t)}[k]} \sum \left\{ \theta_i^{(t)}[k] | \mathcal{I}_i^{(t)}[k] = 1 \right\}$
      if $k \in [N_d]$ **then**
        Deactivate k for clients with smaller gradients
      end if
    end for
    Deactivate clients that have less activated parameters
    Reactivation$(\mathcal{D}_A, \mathcal{D}_A^{(t)}, \mathcal{D}_A^{(t+1)})$
  end for

ClientUpdate(i, I_i, θ)
  $\mathcal{B} \leftarrow$ (split training data on H_i into batches of size B)
  for each local epoch e from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $\theta \leftarrow \theta - \eta \nabla(\theta; b)$
    end for
  end for
  return $\{\theta_i^{(t)}[k] | \mathcal{I}_i^{(t)}[k] = 1\}$

Meanwhile, based on the collected gradients, the server updates its $\mathcal{I}^{(t)}$ to $\mathcal{I}^{(t+1)}$ by evaluating the relative contribution of each involved D_i in round t. The main idea of the evaluation is that if a client D_i's contribution for parameters $\theta[k]$ in round t is relatively trivial, the server will not request the k-th parameters from D_i in round $t+1$. Note that the evaluation metric can be any kind of measurements for the contributions of clients. In this work, as discussed in Sec.5.3, we set a threshold as the mean value of all returned gradients, and construct the updating function as follows:

$$\mathcal{I}_i^{(t+1)}[k] = \begin{cases} 0, & \mathcal{I}_i^{(t)}[k] = 1 \text{ and } \theta^{(t+1)}[k] > \theta_i^{(t)}[k], \\ \mathcal{I}_i^{(t)}[k], & \text{otherwise.} \end{cases} \quad (7)$$

5.4.2 Reactivation of Clients and Parameters. As described in Sec.5.2 and Sec.5.3, there are two different implementation strategies for reactivating clients and parameters in FedDA. *Restart* re-initializes the process whenever there are less than β_r of total clients to be involved in the next round, and *Explore* randomly explores the deactivated clients and ensures there are at least β_e of total clients being activated in each round. Complete pseudo-code for these two strategies are given in Algorithm 2 and Algorithm 3.

5.4.3 Communication Efficiency Analysis. Here we present a mathematical efficiency analysis of our proposed framework FedDA. Suppose the expectation of the portion of remaining clients after each round is r_c and the portion of deactivated parameters is r_p. According to the following analysis, with r_c adn r_p smaller than

Algorithm 2 *Restart* strategy for reactivating clients.

Reactivation($\mathcal{D}_A, \mathcal{D}_A^{(t)}, \mathcal{D}_A^{(t+1)}$)
  if $|\mathcal{D}_A^{(t+1)}| < \beta_r M$ **then**
    $\mathcal{D}_A^{(t+1)} \leftarrow \mathcal{D}$
    $\mathcal{I}^{(t+1)} \leftarrow [1]$
  end if
  return $\mathcal{D}_A^{(t+1)}, \mathcal{I}^{(t+1)}$

Algorithm 3 *Explore* strategy for reactivating clients.

Reactivation($\mathcal{D}_A, \mathcal{D}_A^{(t)}, \mathcal{D}_A^{(t+1)}$)
  if $|\mathcal{D}_A^t| < \beta_e M$ **then**
    sample $(\beta_e M - |\mathcal{D}_A^{(t)}|)$ clients from $\mathcal{D} \setminus \mathcal{D}_A^{(t)}$ to rejoin $\mathcal{D}_A^{(t+1)}$
  end if
  return $\mathcal{D}_A^{(t+1)}, \mathcal{I}^{(t+1)}$

1, FedDA can indeed enhance the communication efficiency as expected.

For *Restart* strategy, the expectation of the amount of communicated parameters for each iteration before reinitializing can be calculated as

$$\mathbb{E}[\#cp] = MN\frac{1-r_c^{t_0+1}}{1-r_c} - MN_d\frac{r_c r_p - (r_c r_p)^{t_0+1}}{1-r_c r_p}, \tag{8}$$

where t_0 is the number of expected rounds before restarting, which satisfies the equation $t_0 \geq log_{r_c}\beta_r$. Thus, the expected number of communicated parameters comparing with vanilla FedAvg is

$$\frac{\mathbb{E}[\#cp]}{t_0 MN} = \frac{1-r_c^{t_0+1}}{1-r_c} - \frac{N_d}{N}\frac{r_c r_p - (r_c r_p)^{t_0+1}}{1-r_c r_p}. \tag{9}$$

Similarly, for *Explore* strategy, the expectation of the amount of communicated parameters for each round starting from the second round is

$$\begin{aligned}\mathbb{E}[\#cp] = {} & M\beta_e r_c \gamma (N - r_p N_d) \\ & - M\beta_e r_c (1-\gamma)(N - \hat{r_p} N_d) \\ & + MN\beta_e (1 - r_c),\end{aligned} \tag{10}$$

where γ is the portion of clients that has been in the activated list before the last round, and $\hat{r_p}$ is their expected portion of deactivated parameters. Obviously, $\hat{r_p} \geq r_p$, which means the communication costs comparison starting from the second round can be bounded as

$$\frac{\mathbb{E}[\#cp]}{t_0 MN} \leq \beta_e - \beta_e r_c r_p \frac{N_d}{N}. \tag{11}$$

6 EXPERIMENTS

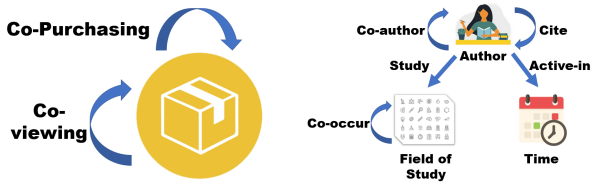
We conduct comprehensive experiments on two real-world heterographs constructed from open benchmark datasets in two application scenarios. We compare the models towards the practical link prediction task and conduct in-depth analysis to illustrate the advantages of our proposed techniques. To fully demonstrate the superiority of our model, we conduct experiments to verify the following four research questions (RQs):

- **RQ1** Compared with FedAvg, does FedDA achieve better performance in the downstream task?
- **RQ2** Can the FedDA enhance communication efficiency as expected?
- **RQ3** How does FedDA converge on real data comparing with FedAvg and global training?
- **RQ4** How does the setting of hyper-parameters (such as α and β) affect FedDA?

6.1 Experimental settings

We use two commonly studied datasets, including one used in the original paper of S-HGN [20], to study the performance of our proposed FedDA frameworkStatistics of the datasets are shown in Table 1, and the schemas of these two heterographs are shown in Fig.4. Link prediction on these datasets is common and challenging [2, 8, 15, 25].

- **Amazon** is a dataset for online purchasing. Following S-HGN, we use the subset proposed by GATNE [2], which contains product metadata of electronics categories, with co-viewing and co-purchasing links. The node features are 1156-dim vectors generated by the price, sales-rank, brand, and category. The schema of this e-commerce heterograph is shown in 4(a).
- **DBLP** is a citation network including nodes of authors, venues, years and papers. We use a subgraph of the dataset generated by HNE [41]. Specifically, phrases are extracted by the AutoPhrase algorithm from paper texts, and their features are the aggregation of 300-dim word embeddings generated by word2vec on all paper texts. Author and venue features are the aggregations of their corresponding paper features, which are also calculated based on word embeddings. In this work, we consider the subgraph containing all authors who have publications in **International Conference on Data Engineering**, the years those authors are active in and the phrases they study. The schema of this bibliographic heterograph is shown in Fig. 4(b).



(a) Amazon (b) DBLP

Figure 4: Schemas of heterographs constructed from Amazon and DBLP data.

Table 1: Statistics of the datasets.

Dataset	#Nodes	#Node Types	#Edges	#Edge Types	Density
Amazon	10,099	1	148,659	2	0.15%
DBLP	114,145	3	7,566,543	5	0.58%

System synthesis. The edges on the global graph of Amazon is splited into training set (90%) and test set (10%), whereas the edges on the global graph of DBLP is splited into training set (85%) and

test set (15%). As for simulating the distributed system, we split both datasets through random sampling. Specifically, every client will first randomly select the types of edges they are specialized with, and sample r_a of them out of the global graph. Then, for the rest of the types, they sample r_b of them, which is set to be a much smaller value than r_a. In our experiments, we set r_a to be 30% and r_b to be 5%. Note that the biased clients only perform link prediction with edges they are specialized with, but for global test data, the downstream link prediction task is performed over all types of links.

Since our proposed FedDA framework can fit any HGN model, we just use the default structure of S-HGN, which is a three-layer model with three-head attention weights. The learning rate is set to 0.0005. We find the best-performed hyper-parameters by grid search which will be further discussed in Sec.6.2, and find the best β_r for *Restart* strategy to be 0.4, β_e for *Explore* strategy to be 0.667, and α for both strategies to be 0.5. We implement our model on top of the code from [20] which is based on pytorch[3] library. The experiments are executed on server with 8 NVIDIA GeForce GTX 1080 Ti GPUs. All code and data are provided in this GitHub repository with clear instructions[4].

Baselines and Evaluation Metrics We conduct comprehensive evaluations by comparing S-HGN under FedDA with three natural baseline frameworks, i.e., 1) Global model: S-HGN trained on the original global heterograph without data partitioning, which is supposed to be an upper bound for any FL framework without consideration of the missing links and lack of node alignment between clients, 2) Local model: S-HGN trained solely on each client locally (with performance averaged), which is presented as a lower bound without any consideration from the global view, 3) FedAvg: S-HGN trained collaboratively across all clients under the vanilla FedAvg framework. Following previous work, we evaluate the performance with the metrics of ROC-AUC (**A**rea **U**nder the **ROC** **C**urve) and MRR (**M**ean **R**eciprocal **R**ank), which are the two common metrics for link prediction on heterographs. As for communication efficiency, we report the amount of total transmitted parameters of two FL frameworks. For global training and FL frameworks, we report the average performance over five runs, while for local models, the scores are further averaged across all models trained locally on each client.

6.2 Experimental results and analysis

FedDA Effectiveness (RQ1):

The first thing to notice is the great gaps between locally trained and globally trained models, which indicates the potential enhancement if we can properly use the global information gathered through FL. As it turns out, the FL trained models in the lower part of Tab.2 indeed achieve much better performances than local models, which also proves the benefits brought by the collaboration across clients. Furthermore, when comparing FedDA with FedAvg, FedDA constantly achieves better performance. Particularly, the *Explore* strategy tends to outperform the others on DBLP, while *Restart* strategy is more effective for Amazon. Although such gaps are reduced on Amazon due to the limited number of link types to perform the prediction for Non-IID clients. Also, it is worth pointing out that FedDA achieves these better performances with much less communication rounds, activated clients and transmitted parameters, which will be further discuss in the following RQs.

Communication Efficiency (RQ2): We compare the amount of total transmitted gradients of FedDA with FedAvg in Tab.3. Note that to be consistent with Tab.2 and a fair comparison for FedAvg, we are presenting the results after 40 communication rounds for all frameworks after which FedAvg is converged to its global optimum, although FedDA often converges faster than FedAvg as we will discuss in the next RQ. Even so, according to Tab.3, both implementation strategies of FedDA can reduce a significant amount of transmitted parameters, which further proves that the dynamic activation mechanism is working in reducing the communication cost as expected. Note that, although we only computed the amount of transmitted gradients as a measure of efficiency, less transmitted gradients naturally means less computation on clients (especially the deactivated ones) and the server.

Convergence Studies (RQ3): Take S-HGN working on both dataset with 16 clients as an example, we present the convergence curves in Fig.5. Notably, *Restart* and *Explore* can both achieve better performance with significantly less communication rounds. For instance, FedDA with *Explore* on DBLP can achieve a score over 0.537 within 20 rounds, which will take FedAvg 40 rounds. In other words, if we limit the number of communication rounds for both frameworks, FedDA would achieve a much better performance. Moreover, if we take the number of necessary communication rounds into consideration while calculating the transmitted costs, the efficiency enhancement would become even more significant. For instance, if we run FedDA with *Explore* strategy for only 20 rounds on DBLP with 16 clients, we will have a model just as effective as FedAvg with 40 rounds, saving approximately 75% transmitted parameters. Moreover, if we consider the best and the worst performance of these frameworks like in Sec.4, it appears that FedDA could also enhance the minimum score comparing with FedAvg activating all clients as shown in Fig.5(d) and Fig.5(c), which further proves the dynamic activation strategy is working in stabilizing the FL process as expected.

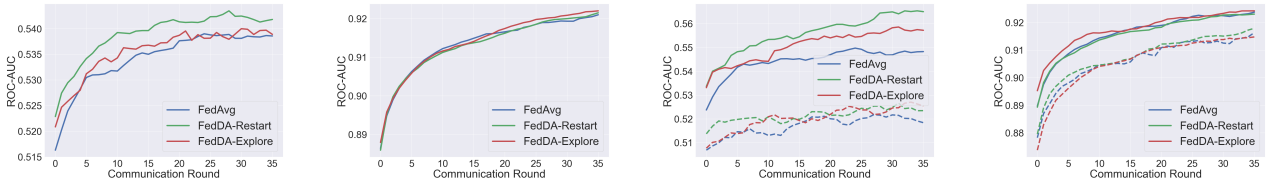
Hyper-parameter Studies (RQ4): We compare the link prediction performance regarding the ROC-AUC curves generated by FedDA with different α, β_e and β_r. Results are shown in Fig. 6. In Fig.6(a), we fix other parameters and study β_r for *Restart* strategy, while for Fig.6(b) and Fig.6(c) we present *Explore* strategy with different α and β_e. All the results are generated by training S-HGN on DBLP with 16 clients.

Observed from Fig. 6(a), β_r affects the final performance significantly. Thus, choosing a proper β_r, which controls the amount of communication rounds before re-initializing, is crucial to elevate the final testing accuracy. Furthermore, there appears to be a trade-off between communication efficiency and final model performance. That is, if the total number of communication rounds is fixed, then a smaller β_r will tolerate less clients being activated before reactivating them all, leading to better communication efficiency while sacrificing final performance. As for α in Fig.6(b), it seems a too

[3] https://pytorch.org/
[4] https://github.com/dongzizhu/FedDA

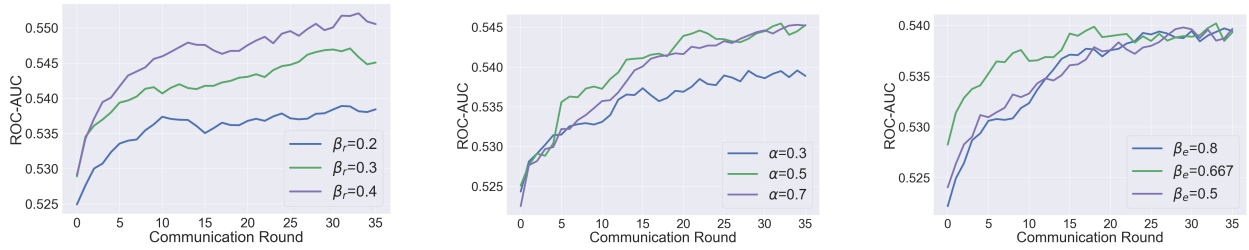
Table 2: Link prediction results on two datasets with varying numbers of clients. Besides averaged accuracy, we also provide the corresponding std. FedDA [1] reactivate clients with *Restart* strategy, FedDA [2] reactivate clients with *Explore* strategy.

Framework	DBLP						Amazon			
	ROC-ACU			MRR			ROC-ACU		MRR	
Global	0.7750 ± 0.0087			0.9015 ± 0.0045			0.9215 ± 0.0046		0.9604 ± 0.0035	
Local	0.4980 ± 0.0062			0.7503 ± 0.0045			0.7522 ± 0.0945		0.9066 ± 0.0496	
	M=4		M=8		M=16		M=8		M=16	
	ROC-ACU	MRR	ROC-ACU	MRR	ROC-ACU	MRR	ROC-ACU	MRR	ROC-ACU	MRR
FedAvg	0.5480 ± 0.0081	0.7804 ± 0.0031	0.5309 ± 0.0044	0.7792 ± 0.0065	0.5382 ± 0.0074	0.7710 ± 0.0085	0.9200 ± 0.0054	0.9661 ± 0.0022	0.9187 ± 0.0033	0.9655 ± 0.0029
FedDA [1]	0.5379 ± 0.0025	0.7753 ± 0.0063	0.5443 ± 0.0064	**0.7807 ± 0.0061**	**0.5422 ± 0.0093**	**0.7822 ± 0.0082**	**0.9202 ± 0.0023**	**0.9666 ± 0.0027**	0.9201 ± 0.0051	**0.9668 ± 0.0032**
FedDA [2]	**0.5504 ± 0.0087**	**0.7865 ± 0.0093**	**0.5450 ± 0.0056**	**0.7807 ± 0.0049**	0.5407 ± 0.0042	0.7784 ± 0.0070	0.9200 ± 0.0054	0.9660 ± 0.0022	**0.9203 ± 0.0033**	0.9645 ± 0.0021



(a) DBLP (b) Amazon (c) DBLP (d) Amazon

Figure 5: Training curves of compared frameworks. In Fig.5(a) and Fig.5(b), the curves are average performance of five runs. In Fig.5(c) and Fig.5(d), the solid lines are the highest score over test data in each round, while the dotted lines are the lowest score. All curves are generated with 16 clients.



(a) β_r for *Restart* Strategy (b) α for *Explore* Strategy (c) β_e for *Explore* Strategy

Figure 6: Training curves of FedDA with different parameters settings.

Table 3: Average total amount of transmitted gradients on two datasets with varying numbers of clients.

Framwork	DBLP			Amazon	
	M=4	M=8	M=16	M=8	M=16
FedAvg	10,400	20,800	41,600	133,40	26,680
FedDA [1]	8,884	14,687	29,699	9,180	18,637
FedDA [2]	6,587	14,105	30,134	8,011	17,290

small α will lead to an unstable training process and worse performance. Maybe it's because the server tend to deactivate clients more aggressively with larger α, which means it will repeatedly activate clients with all of their parameters and thus lead to better exploration of the global information. At last, intuitively, the smaller β_e, the more transmitted parameters we can save. However, in this work, we only consider efficient models with the best performance. Thus we regard $\beta_e = 0.667$ as the best setting, since it is the most effective model according to Fig.6(c).

7 CONCLUSION

In this work, we study the demanded yet challenging setting of FL across distributed Non-IID heterographs. We revisit the vanilla FedAvg protocol over heterographs and surprisingly discover that the partial activation of clients and parameters can potentially benefit the final model's performance and communication efficiency when the link types in sub-heterographs are Non-IID. To reduce activated clients and parameters while addressing the instability in performance under random activation, we design a dynamic activation protocol FedDA, which is able to adaptively evaluate clients and their respective parameters for their strategical selection along the global model's FL training process. Comprehensive empirical analyses corroborate the effectiveness of our proposed techniques in reducing the communication cost while stabilizing the FL process. Further studies on privacy and fairness issues on top of FedDA, as well as its potential connection with generic FL beyond graph data can both be interesting future directions.

REFERENCES

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*.

[2] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network. *KDD* (2019).

[3] Chuan Chen, Weibo Hu, Ziyue Xu, and Zibin Zheng. 2021. FedGL: Federated Graph Learning Framework with Global Self-Supervision. arXiv:2105.03170 [cs.LG]

[4] Chong Chen, Weizhi Ma, Min Zhang, Zhaowei Wang, Xiuqiang He, Chenyang Wang, Yiqun Liu, and Shaoping Ma. 2021. Graph Heterogeneous Multi-Relational Recommendation. In *TNNLS)*.

[5] Fahao Chen, Peng Li, Toshiaki Miyazaki, and Celimuge Wu. 2022. FedGraph: Federated Graph Learning With Intelligent Sampling. *IEEE Transactions on Parallel and Distributed Systems (TPDS)* (2022).

[6] Mingyang Chen, Wen Zhang, Zonggang Yuan, Yantao Jia, and Huajun Chen. 2021. Fede: Embedding knowledge graphs in federated setting. In *International Joint Conference on Knowledge Graphs (IJCKG)*. 80–88.

[7] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *ICLR*.

[8] Daniel Cummings and Marcel Nassar. 2020. Structured citation trend prediction using graph neural networks. In *The International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

[9] Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Carl Yang, Han Xie, Lichao Sun, Lifang He, Liangwei Yang, Philip S. Yu, Yu Rong, Peilin Zhao, Junzhou Huang, Murali Annavaram, and Salman Avestimehr. 2021. FedGraphNN: A Federated Learning System and Benchmark for Graph Neural Networks. arXiv:2104.07145 [cs.LG]

[10] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. 2020. The non-iid data quagmire of decentralized machine learning. In *ICML*.

[11] Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. 2021. Mixgcf: An improved training method for graph neural network-based recommender systems. In *KDD*.

[12] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *TNNLS* (2021).

[13] Zhihao Jia, Sina Lin, Mingyu Gao, Matei Zaharia, and Alex Aiken. 2020. Improving the accuracy, scalability, and performance of graph neural networks with roc. *Proceedings of Machine Learning and Systems* 2 (2020), 187–198.

[14] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*.

[15] Song Jiang, Bernard Koch, and Yizhou Sun. 2021. HINTS: Citation Time Series Prediction for New Publications via Dynamic Heterogeneous Information Network Embedding. In *WWW*.

[16] Justin M Johnson and Taghi M Khoshgoftaar. 2019. Survey on deep learning with class imbalance. *Journal of Big Data* 6 (2019), 1–54.

[17] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. 2021. A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection. (2021).

[18] Ji Liu, Jizhou Huang, Yang Zhou, Xuhong Li, Shilei Ji, Haoyi Xiong, and Dejing Dou. 2022. From distributed machine learning to federated learning: A survey. *Knowledge and Information Systems* (2022), 1–33.

[19] Linhao Luo, Yixiang Fang, Xin Cao, Xiaofeng Zhang, and Wenjie Zhang. 2021. Detecting Communities from Heterogeneous Graphs: A Context Path-based Graph Neural Network Model. In *CIKM*.

[20] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. 2021. Are we really making much progress? Revisiting, benchmarking, and refining heterogeneous graph neural networks. *KDD* (2021).

[21] Lingxiao Ma, Zhi Yang, Youshan Miao, Jilong Xue, Ming Wu, Lidong Zhou, and Yafei Dai. 2019. {NeuGraph}: Parallel Deep Neural Network Computation on Large Graphs. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. 443–458.

[22] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[23] Guangxu Mei, Ziyu Guo, Shijun Liu, and Li Pan. 2019. SGNN: A Graph Neural Network Based Federated Learning Approach by Hiding Structure. In *IEEE International Conference on Big Data (ICBD)*.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *NeurIPS* (2013).

[25] Barbara Plank and Reinard van Dalen. 2019. CiteTracked: A Longitudinal Dataset of Peer Reviews and Citations.. In *BIRNDL@ SIGIR*.

[26] Morteza Ramezani, Weilin Cong, Mehrdad Mahdavi, Mahmut T Kandemir, and Anand Sivasubramaniam. 2021. Learn Locally, Correct Globally: A Distributed Algorithm for Training Graph Neural Networks. *arXiv preprint arXiv:2111.08202* (2021).

[27] Mrinmaya Sachan and Eric Xing. 2016. Easy questions first? a case study on curriculum learning for question answering. In *ACL*.

[28] Shreyas Saxena, Oncel Tuzel, and Dennis DeCoste. 2019. Data parameters: A new family of parameters for learning a differentiable curriculum. *Advances in Neural Information Processing Systems* 32 (2019).

[29] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. 2016. A survey of heterogeneous information network analysis. *TKDE* 29 (2016), 17–37.

[30] Elena Sügis, Jerome Dauvillier, Anna Leontjeva, Priit Adler, Valerie Hindie, Thomas Moncion, Vincent Collura, Rachel Daudin, Yann Loe-Mie, Yann Herault, et al. 2019. HENA, heterogeneous network-based data set for Alzheimer's disease. *Scientific data* 6 (2019), 1–18.

[31] Alok Tripathy, Katherine Yelick, and Aydın Buluç. 2020. Reducing communication in graph neural network training. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–14.

[32] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. *ICLR* (2018).

[33] Marc Vidal, Michael E Cusick, and Albert-László Barabási. 2011. Interactome networks and human disease. *Cell* 144 (2011), 986–998.

[34] Cheng Wan, Youjie Li, Cameron R Wolfe, Anastasios Kyrillidis, Nam Sung Kim, and Yingyan Lin. 2022. PipeGCN: Efficient full-graph training of graph convolutional networks with pipelined feature communication. *arXiv preprint arXiv:2203.10428* (2022).

[35] Xin Wang, Yudong Chen, and Wenwu Zhu. 2021. A survey on curriculum learning. *TPAMI* (2021).

[36] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, and Bryan Hooi. 2021. Curgraph: Curriculum learning for graph classification. In *WWW*.

[37] Zhaonan Wang, Tianqi Xia, Renhe Jiang, Xin Liu, Kyoung-Sook Kim, Xuan Song, and Ryosuke Shibasaki. 2021. Forecasting Ambulance Demand with Profiled Human Mobility via Heterogeneous Multi-Graph Neural Networks. In *ICDE*. 1751–1762.

[38] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. FedGNN: Federated Graph Neural Network for Privacy-Preserving Recommendation. arXiv:2102.04925 [cs.IR]

[39] Han Xie, Jing Ma, Li Xiong, and Carl Yang. 2021. Federated graph classification over non-iid graphs. In *NeurIPS*.

[40] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. 2020. Heterogeneous network representation learning: A unified framework with survey and benchmark. *TKDE* (2020).

[41] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. 2020. Heterogeneous Network Representation Learning: A Unified Framework with Survey and Benchmark. *TKDE* (2020).

[42] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. 2019. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13 (2019), 1–207.

[43] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. 2021. Subgraph federated learning with missing neighbor generation. In *NeurIPS*.

[44] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. 2021. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Web Search and Data Mining (WSDM)*.

[45] Da Zheng, Chao Ma, Minjie Wang, Jinjing Zhou, Qidong Su, Xiang Song, Quan Gan, Zheng Zhang, and George Karypis. 2020. Distdgl: distributed graph neural network training for billion-scale graphs. In *2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3)*. IEEE, 36–44.

[46] Jun Zhou, Chaochao Chen, Longfei Zheng, Huiwen Wu, Jia Wu, Xiaolin Zheng, Bingzhe Wu, Ziqi Liu, and Li Wang. 2021. Vertically Federated Graph Neural Network for Privacy-Preserving Node Classification. arXiv:2005.11903 [cs.LG]

[47] Rong Zhu, Kun Zhao, Hongxia Yang, Wei Lin, Chang Zhou, Baole Ai, Yong Li, and Jingren Zhou. 2019. AliGraph: A Comprehensive Graph Neural Network Platform. *Proc. VLDB Endow.* (2019), 2094–2105.

[48] Xinghua Zhu, Jianzong Wang, Zhenhou Hong, and Jing Xiao. 2020. Empirical studies of institutional federated learning for natural language processing. In *EMNLP*.