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

Zishan Gu
Department of Computer Science
Columbia University
New York, United States
zg2409@columbia.edu

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

Guangji Bai
Department of Computer Science
Emory University
Atlanta, United States
guangji.bai@emory.edu

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

Liang Zhao
Department of Computer Science
Emory University
Atlanta, United States
liang.zhao@emory.edu

Carl Yang*
Department of Computer Science
Emory University
Atlanta, United States
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 biased local 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. Moreover, we design a novel disentangled model D-HGN to enable type-oriented activation of model parameters for FedDA. The effectiveness and efficiency of our proposed techniques are backed by both theoretical and empirical analysis– We theoretically analyze the validity and convergence of FedDA and 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.**

***Index Terms*—heterogeneous graphs, federated learning, dynamic activation, performance gain, efficiency gain**

I. Introduction

Heterogeneous graphs constructed with multiple types of nodes and links can well record and model complex real-world application scenarios [1], such as citation prediction on bibliographical networks [2]–[4], patient profiling on healthcare networks [5]), emergency medical service [6] and recommender systems [7]–[9]. 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 heterogeneous graph.

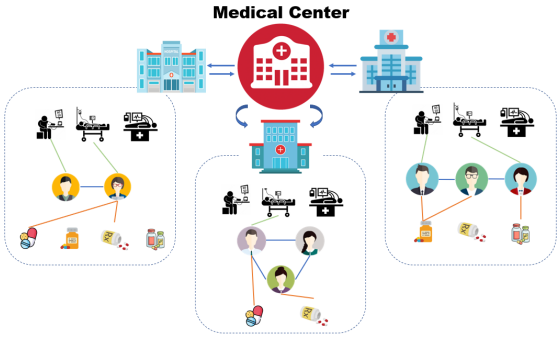


Fig. 1. A toy example of the real-world clinical heterogeneous graph.

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 heterogeneous graph, it can be regarded as a non-independent-identically-distributed (Non-IID) heterogeneous subgraph of the entire global heterogeneous graph containing all potential healthcare data generated

*Corresponding author

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

Federated learning (FL) [10] 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 CV and NLP, FL exhibits enormous potential in effectiveness, especially when data are IID across data owners [11]. Recently, FL also achieves remarkable progress in tackling relational data mining tasks, *i.e.*, for graph learning [12]–[16]. 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 heterogeneous graphs, we first conduct a preliminary analysis to examine the pivotal factors that determine the utility of the outcome graph learning model. For simplicity, we use the state-of-the-art heterogeneous graph model of Simple-HGN [17] with the FedAvg protocol [18] 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 heterogeneous graphs.

Challenge 1: *How to dynamically activate clients and parameters to obtain the final generalized global model?*

Solution 1: Vanilla 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.

Challenge 2: *Can we leverage the characteristics of heterogeneous graphs and heterogeneous graph neural networks (HGNs) to further excite the potential of FedDA?*

Solution 2: FedDA with D-HGN Activating parameters solely based on gradients can lead to unwanted instability in FL. To this end, we propose to enable type-oriented activation of graph model parameters for FedDA by designing a novel disentangled model D-HGN. For an input heterogeneous ego-graph, at each hop, D-HGN models each link type by employing an exclusive set of parameters. Hence, in an activated client, parameters responsive to modeling a certain link type can be precisely activated or vice versa. Applying such disentangled model leads to improved convergence and stability in the FL training process on distributed heterogeneous graphs.

Approaches We provide two strategies to implement the dynamic activation of clients and parameters in FedDA towards improved performance and efficiency. Based on these strategies, theoretical analysis is provided on their correctness and convergence, whereas mathematical analysis is conducted towards their saves on computational costs.

To empirically verify the utility of our proposed methods, we perform experiments on multiple distributed heterogeneous graph systems realistically synthesized four real-world benchmark datasets of heterogeneous graphs from different application scenarios. Experimental results indicate that FedDA can lead to a global model with significantly improved performance and efficiency, which are further boosted by the employment of D-HGN. In-depth training analysis and hyper-parameter studies further corroborate the improved convergence, stability, and robustness of FedDA.

II. Related Works

A. Federated learning on graphs

Federated learning (FL) [19], [20] 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, Zhou et al. [21] and Mei et al. [22] studied the scenarios where node features and structures on distributed graphs vary across local devices. In the more common horizontal FL setting, He et al. [14] proposed an open-source benchmark system for federated GNNs; and there are also some works studied the distributed homograph setting under the consideration of cross-graph links [12], [15], [16]. Moreover, Xie et al. [23] considered latent heterogeneous link distributed in the FL setting, but their graphs are still on homogeneous.

With the consideration of multiple types of links, Wu et al. [24] studied FL over the bipartite user-item graph for recommender systems; Chen et al. [25] studied FL over knowledge graphs for their completion. In complex real-world applications, heterogeneous graphs that include multiple types of both nodes and links can better simulate and model realistic networks compared to knowledge graphs [1], [26]. The superior richness of information in heterogeneous graphs 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.

As for dealing with Non-IID data in the FL framework, Li et al. [27] proposed FedProx to tackle clients' heterogeneity through a generalization and re-parametrization of FedAvg. Wang et al. [28] utilized model-agnostic meta-learning to handle Non-IID graph data while preserving the model generalizability. GCFL [13] dynamically finds clusters of local

systems based on the gradients to perform graph classification through FL across data from different domains. However, these methods are either not specifically designed for tasks over graphs or not suitable for tasks on heterogeneous graphs. To the best of our knowledge, our work is the very first attempt towards effective FL over distributed Non-IID heterogeneous graphs. Besides improving the performance compared to basic FL, our proposed approach further steps out to uplift the communication efficiency.

B. Strategic training of GNNs

Similar to traditional machine learning models, when the testing heterogeneous graph is considered as a generic graph following the global data distribution, training a GNN on biased (Non-IID) local heterogeneous graphs degenerates its test performance [29], [30]. Several training strategies designed for CV and NLP [31]–[34] tasks in solving the Non-IID problem have been transferred to graph learning tasks.

For example, inspired by cognitive science, curriculum learning [35] points out the pivotal role of training sample selection in machine learning tasks, which is now applied in graphs as well. Wang et al. [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 [37]. Huang et al. [38] 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. 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 cannot.

III. Problem Formulation

System We denote a global heterogeneous graph 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. Each $e \in E$ is associated with an edge type $\psi(e)$, which is determined by the types of nodes on its two ends. In this work, we consider heterogeneous graphs 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 heterogeneous subgraph $\mathcal{H} = \{H_i | i \in [M]\}$. 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 heterogeneous graph 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_* as the probability $\mathcal{P} = P(\psi(e) | e \in E_*)$. For any two different clients D_i and D_j, we have $\mathcal{P}_i \not\sim \mathcal{P}_j$, which we refer to as biased data regarding Non-IID link types in distributed heterogeneous graph 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 heterogeneous graphs [17]. Technically, for a pair of nodes on the heterogeneous graphs, link prediction infers whether there exits an edge between them. Therefore, we formulate our goal of federated link prediction on heterogeneous graphs 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 heterogeneous graph 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

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

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 [39]). For the complex and highly irregular heterogeneous graphs, 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

heterogeneous graphs to find out pivotal factors in FedAvg that determine the outcome graph learning model's utility.

IV. Motivating Preliminary Studies

Here, we present intuitive preliminary studies to investigate the problem of heterogeneous graph FL across the distributed system. Specifically, we adopt the widely used weights-sharing-based FL protocol, FedAvg [18], with the state-of-the-art heterogeneous graph model of Simple-HGN [17] on a small subgraph from the benchmark heterogeneous network of DBLP [40]. 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 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 for the traditional FedAvg framework when applied to the distributed heterogeneous graphs?

A. Simple-HGN

Lv et al. [17] studied 12 recent state-of-the-art heterogeneous graph neural networks (HGNs) and found that vanilla GAT [41], a homogeneous GNN, surprisingly outperforms all compared HGNs. Based on the finding, they proposed a simple yet powerful baseline Simple-HGN (S-HGN) by adapting GAT to heterogeneous graphs. Specifically, S-HGN enhances GAT by equipping it with three components: learnable type-wise edge embedding, residual connections between layers, and L_2 normalization on the final output. The structure of the encoder part of S-HGN is shown as the backbone in Fig. 4, while the highlighted part is its extension to Disentangled-HGN (D-HGN), which will be discussed later. S-HGN follows a common encoder-decoder architecture to conduct link prediction, and as shown in [17], it is empirically superior to other compared models.

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 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]:

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

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

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

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

where $\alpha_{uv}^{(l)}$ is the attention weight over edge e between node u and node v , and the activation function here is ELU [42] by default. Note that $W_{res}^{(l)}$ is only needed when the dimensions of hidden features change across layers, and the final output embedding is the concatenation of embeddings from all layers.

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 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 these three enhancements, accompanied by the multi-head attention mechanism of vanilla GAT, S-HGN is claimed to achieve the best performance on heterogeneous graphs with link prediction task. However, its only adoption specific to tasks on heterogeneous graph is the edge type embedding, which makes it hard to disentangle the model under federated learning frameworks.

B. FedAvg

FedAvg [18] is an 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:

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

where η_i is the learning rate for the 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 does not own prior knowledge of the distribution of global and local data. Thus we perform the average without discriminative weights as

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

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

C. 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 heterogeneous graphs, more clients involved in each round do 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

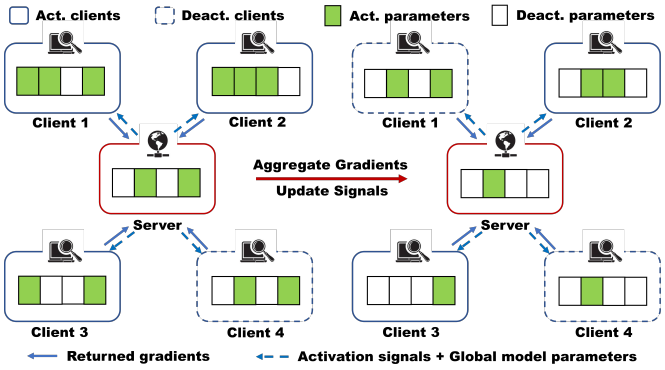


Fig. 3. Illustrative visualization of FedDA. Best viewed in color. The clients in dotted boxes and the parameters in white are the deactivated ones. For each round, the server sends the activation signals and global model to each activated client. The activated clients update the model locally for several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

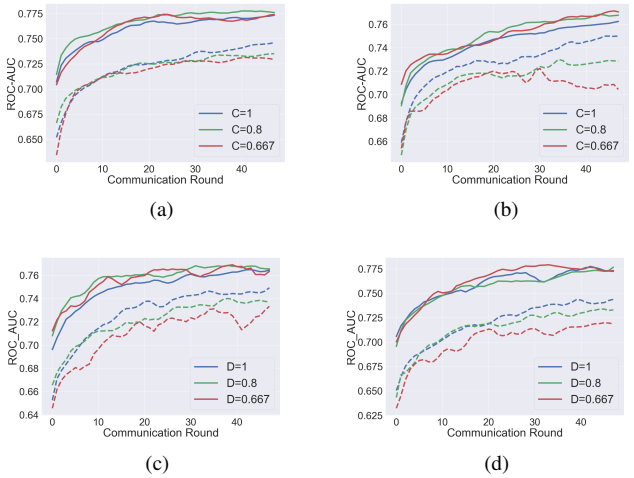


Fig. 2. Performance curves of FedAvg with different client activation rates (C) and parameter activation rates (D). Fig. 2(a) and 2(c) are the results on unbiased data (IID link types) across clients, whereas Fig. 2(b) and 2(d) are the results on biased data (Non-IID link types) as discussed in Sec. III. 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.

version. However, as for the worst models (dashed lines), activating fewer clients may end up with much worse models, especially when edges are Non-IID by their types. That is to say, decreasing the number of involved clients does not harm the optimal performance yet introduces more instability, which implies that we are able to simultaneously enhance the model performance and communication efficiency by activating fewer clients when 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. Herein, 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 the ratio D. As shown in Fig. 2(c) and Fig. 2(d), similarly to activating part of the clients, activating fewer 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 fewer clients involved for each round, while Observation 2 implies that we may be able to gather only a part of the gradients from each client 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 client and parameter selecting strategy for FL to reduce the instability, benefit the data transmission efficiency, while can assist the system in achieving a satisfactory result simultaneously? We leave the discussion of our novel FL framework designs in the following section.

V. 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 Sec. IV-C, we design an 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. We theoretically justify that our FL model's performance loss brought by the reduction of transmitted data during FedDA can be bounded.

In this section, we first discuss the technical details of our proposed FedDA and Disentangled-HGN adapted from S-HGN with theoretical justifications. Then, we discuss the efficiency advantage of FedDA compared with FedAvg.

A. Dynamic Activation of Clients

From our **Observation 1** in Sec. IV-C, 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.

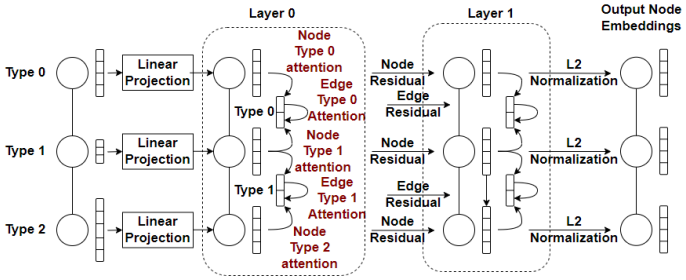


Fig. 4. Structure of Disentangled-HGN.

Intuitively, we want to leave out the 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 the last round. We justify this approximation in the following theorem.

Theorem 1. *[Stationariness of $\mathcal{D}_A$ between steps.] Suppose the neural network is L_f-Lipschitz continuous and L_g-Lipschitz smooth. Given a set of gradients $\{g_k \in \mathbb{R} : k \in \mathcal{I}\}$ which is (ϵ, τ)-separable, i.e., $|g_{k_1} - g_{k_2}| > \epsilon$, $\forall\ k_1 \neq k_2$, and $|\bar{g} - g_k| > \tau\epsilon$, $\forall\ k$, where $\bar{g} \coloneqq (1/|\mathcal{I}|)\sum_{k=1}^{|\mathcal{I}|} g_k$ and $\tau \in (0, \frac{1}{2})$. Then, after one step of gradient descent with stepsize η, the new gradients $\{g'_k \in \mathbb{R} : k \in \mathcal{I}\}$ satisfy*

$$(g'_i - \bar{g}') \cdot (g_i - \bar{g}) > 0, \quad \forall\ k \in \mathcal{I} \tag{6}$$

as long as $\eta < \left(\left(\sqrt{|\mathcal{I}|} + \sqrt{|\mathcal{I}|^{-1}}\right) L_f L_g\right)^{-1} \cdot \tau\epsilon$.

Theorem 1 above guarantees that, with an appropriately small enough stepsize of gradient descent, if the gradient is smaller or greater than the average, the same inequality still holds over the updated gradients after the gradient descent.

Furthermore, if we continuously deactivate clients and update $\mathcal{D}_A$ along with training, the number of involved clients may be reduced rapidly, and the training will be much more 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 follows.

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

B. Dynamic Activation of Parameters

From our **Observation 2** in Sec. IV-C, 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 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 performs link prediction on one single type of links, then the weights in the decoder of S-HGN for other types of links will not be properly trained, which means if we still naively take the average for all parameters without discrimination, the random initialization of those untrained parameters will always be a part of the aggregation and thus lead to poor final performance. 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.V-A 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 $\sum_{k \in [N_d]} \mathcal{I}_i^{(t)} < \alpha \cdot N_d$, $\mathcal{D}_A^{(t+1)} = \mathcal{D}_A^{(t)} \setminus \{D_i\}$.

Disentangled-HGN Since the node and link type distributions can be biased across the Non-IID local heterogeneous subgraphs, it is intuitive to allow clients with more data of certain

[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**
      $\theta_i^{(t)} \leftarrow$ ClientUpdate$(i, I_i^{(t)}, \theta^{(t)})$
    end for
    for parameter $k \in [N]$ **do**
      $\theta^{(t+1)}[k] = \frac{1}{\sum_{D_i \in \mathcal{D}_A^{(t)}} I_i^{(t)}[k]} \sum \{\theta_i^{(t)}[k] | \mathcal{I}_i^{(t)}[k] = 1\}$
      if $k \in [N_d]$ **then** ▷ Deactivation of parameters
        for client $i \in D_A^{(t)}$ **do**
          if $I_i^{(t)}[k] = 1$ and $\theta^{(t+1)}[k] > \theta_i^{(t)}[k]$ **then**
            $I_i^{(t+1)} \leftarrow 0$
          else
            $I_i^{(t+1)} \leftarrow I_i^{(t)}$
          end if
        end for
      end if
    end for
    for client $i \in D_A^{(t)}$ **do** ▷ Deactivation of clients
      if $\sum_{k \in [N_d]} \mathcal{I}_i^{(t)} < \alpha \cdot N_d$ **then**
        $\mathcal{D}_A^{(t+1)} = \mathcal{D}_A^{(t)} \setminus \{D_i\}$
      end if
    end for
    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 $\{\theta_i^{(t)}[k] | \mathcal{I}_i^{(t)}[k] = 1\}$

types of nodes and links to contribute more to the training of the corresponding parts of the model. Therefore, to fully excite the potential of dynamic activation of parameters, we propose to disentangle the learnable parameters in the global model according to the node and link types, and develop a novel *Disentangled-HGN (D-HGN)* by extending the original S-HGN structure to a model with both edge-type-aware and node-type-aware attention parameters. Note that, even so, the amount of parameters for D-HGN is still $O(1)$ regarding the number of link types and node types, which is within the same level of complexity compared with S-HGN. As shown in the red part of fig.4, at each layer l, D-HGN assigns a different attention weight for different types of nodes and edges during the attention score calculation process. Thus, Eq.2 is reformulated as

$$\hat{\alpha}_{uv} = \frac{\exp\left(\sigma\left(\vec{a}^T\left[W_{\varphi(u)}\vec{h}_u || W_{\varphi(v)}\vec{h}_v || W_{\psi(e)}\vec{r}_{\psi(e)}\right]\right)\right)}{\sum\limits_{k \in \mathcal{N}_u} \exp\left(\sigma\left(\vec{a}^T\left[W_{\varphi(u)}\vec{h}_u || W_{\varphi(v)}\vec{h}_k || W_{\psi(e)}\vec{r}_{\psi(e)}\right]\right)\right)},$$

where $W_{\varphi(n)}$ is the attention weight for node n with type $\varphi(n)$ and $W_{\psi(e)}$ is the attention weight for edge e with type $\varphi(e)$. In this way, the parameters in the model can be disentangled to take care of the biased data in each client regarding node types, link types, and different hops of neighbors.

C. The FedDA Algorithm

In this section, we introduce the specific algorithms of FedDA, followed by theoretical analysis on convergence and mathematical analysis on efficiency gain. The overall pipeline of FedDA is shown in Algorithm.1.

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

$$\theta^{(t+1)}[k] = \frac{1}{\sum_{D_i \in \mathcal{D}_A^{(t)}} I_i^{(t)}[k]} \sum \left\{\theta_i^{(t)}[k] | \mathcal{I}_i^{(t)}[k] = 1\right\}. \quad (7)$$

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.V-B, we set a threshold as the mean value of all returned gradients, and construct the updating function as follows:

$$\mathcal{I}_i^{(t+1)}[k] = \begin{cases} 0, & \mathcal{I}_i^{(t)}[k] = 1 \text{ and } \theta^{(t+1)}[k] > \theta_i^{(t)}[k], \\ \mathcal{I}_i^{(t)}[k], & \text{otherwise.} \end{cases} \quad (8)$$

2) Reactivation of Clients and Parameters: As described in Sec.V-A and Sec.V-B, 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. We also present theoretical justification for the convergence of these two strategies right after the algorithms.

Theorem 2. *[Convergence of FedDA with Restart strategy.] Suppose Assumptions 1-4 from [43] hold and $L, \mu, \sigma_i, G, \Gamma$ follows the definition therein. If we choose $\xi = L/\mu$, $\beta =$max$\{8\xi, E\}$, and learning rate $\eta_t = 2/(\mu(\beta + t))$. Then, FedDA with Restart strategy satisfies*

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_T)] - \mathcal{L}(\mathbf{w}^*) \leq \frac{\xi}{\beta + TR^{-1} - 1}\left(\frac{2B}{\mu} + \frac{\mu\beta}{2}\mathbb{E}\|\mathbf{w}_1 - \mathbf{w}^*\|\right),$$

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 $\mathcal{D}_A^{(t+1)}, \mathcal{I}^{(t+1)}$

where $B = \sum_{i=1}^{M} p_i^2 \sigma_i^2 + 6L\Gamma + 8(E-1)^2 G^2$ and R is the number of global iterations between two restarting rounds.

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 $\mathcal{D}_A^{(t+1)}, \mathcal{I}^{(t+1)}$

Theorem 3. *[Convergence of FedDA with Explore strategy.] Suppose Assumptions 1-4 from [43] hold and $L, \mu, \sigma_i, G, \Gamma$ follows the definition therein. ξ, β, η_t and B follow the definition in Theorem 2. Then, FedDA with Explore strategy satisfies*

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_T)] - \mathcal{L}(\mathbf{w}^*) \leq \frac{\xi}{\beta + T - 1}\Big(\frac{2B'}{\mu} + \frac{\mu\beta}{2}\mathbb{E}\|\mathbf{w}_1 - \mathbf{w}^*\|\Big),$$

where $B' = B + \frac{4}{K}E^2G^2$.

3) Communication Efficiency Analysis: Here we present a mathematical efficiency analysis of our proposed framework FedDA. We denote the expectation of the fraction of remaining clients after each round is r_c and the fraction of deactivated parameters is r_p.

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

$$\mathbb{E}[\#cp] = MN\frac{1 - r_c^{t_0+1}}{1 - r_c} - MN_d\frac{r_c r_p - (r_c r_p)^{t_0+1}}{1 - r_c r_p}, \quad (9)$$

where t_0 is the number of expected rounds before restarting, which satisfies the equation $t_0 \geq log_{r_c}\beta_r$. Thus, with the fraction of activated clients denoted by C, the expected number of communicated parameters compared with vanilla FedAvg is

$$\frac{\mathbb{E}[\#cp]}{t_0 CMN} = \frac{1 - r_c^{t_0+1}}{(1 - r_c)C} - \frac{N_d}{N}\frac{r_c r_p - (r_c r_p)^{t_0+1}}{(1 - r_c r_p)C}. \quad (10)$$

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

$$\begin{aligned}\mathbb{E}[\#cp] = {} & M\beta_e r_c \gamma(N - r_p N_d) \\ & - M\beta_e r_c(1-\gamma)(N - \hat{r_p} N_d) \\ & + MN\beta_e(1 - r_c),\end{aligned} \quad (11)$$

where γ is the fraction of clients that have been in the activated list before the last round, and $\hat{r_p}$ is their expected fraction 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

$$\frac{\mathbb{E}[\#cp]}{t_0 CMN} \leq \frac{\beta_e}{C} - r_c r_p \frac{N_d}{N}\frac{\beta_e}{C}. \quad (12)$$

Note that, according to our analysis in Sec. IV, when we only activate a fraction of clients in each round, the performance is highly unstable compared with activating all of the clients with Non-IID distributed systems, that is, both the average final performance and the worst final performance suffer from partially activated FedAvg. Moreover, it is well established that, even with IID data, a lower C in FedAvg means the global model will need more communication rounds to converge and reach a target accuracy. Therefore, simply setting C as smaller than 1 in FedAvg is not an ideal solution for improving the efficiency of FL. On the contrary, in this work, we consider the proper dynamic selections of clients and parameters with r_c and r_p smaller than 1, to effectively reduce the computations in FL without significantly harming the performance. Furthermore, from Eq. (10) and Eq. (12), we know larger N_d may lead to more communication efficiency benefits under the same status, which gives our D-HGN model an intrinsic efficiency advantage. We will further discuss this in Sec. VI.

VI. Experiments

We conduct comprehensive experiments on four real-world heterogeneous graphs constructed from open benchmark datasets in two application scenarios. We compare the models towards the practical link prediction task and conduct in-depth analyses 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 compared with FedAvg and global training?
- **RQ4** How does the setting of hyper-parameters (such as α and β) affect FedDA?

A. Experimental settings

We use four commonly studied datasets, including one used in the original paper of S-HGN [17], to study the performance of our proposed FedDA framework and D-HGN model. Statistics of the datasets are shown in Tab. VI-A. Link prediction on tested datasets is common [44]–[47].

- **DBLP** is a citation network including nodes of authors, venues, years, and papers. We use a subgraph of the dataset generated by HNE [40]. Specifically, we consider

the subgraph containing all the authors with publications in **International Conference on Data Engineering**, the years those authors are active in, and the phrases they study.

- **Amazon** is a dataset for online purchasing. Following S-HGN, we use the subset proposed by GATNE [44], which contains product metadata of electronics categories, with co-viewing and co-purchasing links.
- **LastFM** is an online music website. HetRec 2021 [48] first released it to be a graph containing links between users, artists and tags. We use the version released by S-HGN which is preprocessed to leave out the users and tags with only one link.
- **PubMed** is a network of genes, diseases, chemicals, and species constructed from a biomedical literature library. It is first released by HNE with links generated through open relation pattern mining and manual selection.

TABLE I
STATISTICS OF THE DATASETS.

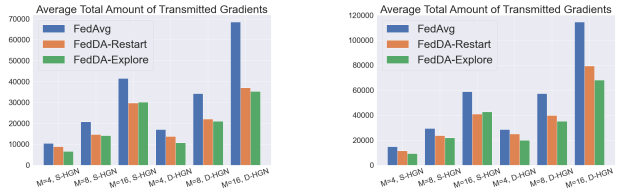
Dataset	$\|V\|$ ($avg(\|V_i\|)$)	#Node Types	$\|E\|$ ($avg(\|E_i\|)$)	#Edge Types	Density
DBLP	114,145 (44,425)	3	7,566,543 (262,858)	5	0.058%
Amazon	10,099 (7774)	1	148,659 (32,039)	2	0.15%
LastFM	20,612 (6,849)	3	141,521 (19,105)	3	0.033%
PubMed	63,109 (19,950)	4	244,986 (32,434)	10	0.0061%

System synthesis The edges on the global graph of Amazon, PubMed and LastFM are split into a training set (90%) and a test set (10%), whereas the edges on the global graph of DBLP are split into a training set (85%) and a test set (15%). The training sets are further split into a training set (90%) and a validation set (10%). As for simulating the distributed system, we split each dataset through random sampling, and the averaged statistics of subsets are shown in Tab. VI-A. 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 a same default structure of S-HGN, which is a three-layer model with three-head attention weights. The dimension of hidden layer is 64, the dimension of edge embedding is 32, the negative slope of LeakyReLU is 0.01, the dropout rate is 0.5 and the weight decay for Adam optimizer is 0.0001. The μ for FedProx is set to be 0.001. We also perform a grid search for learning rate lr since our theoretical analysis points out that FedDA may be sensitive to high learning rate. As it turned out, lr does not effect the final performance much as long as it is smaller than 0.01, which is within the common setting range for learning rates. So, in this work, we simply follow the default setting of S-HGN in its original paper and set the lr to be 0.0005. We use a batch size of 4096 for DBLP, 1024 for Amazon, 8192 for LastFM and PubMed, which are selected on the validation sets. We find the best-performed hyper-parameters for FedDA by grid search which will be further discussed in Sec. VI-B, and find the best β_r for $Restart$ strategy to be 0.2, 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 [17] which is based on pytorch[3] library. The experiments are executed on a 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 and D-HGN under FedDA with four natural baseline frameworks, i.e., 1) Global model: S-HGN and D-HGN trained on the original global heterogeneous graph 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 and D-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 and D-HGN trained collaboratively across all clients under the vanilla FedAvg framework, and 4) FedProx: S-HGN and D-HGN trained collaboratively across all clients under the vanilla FedProx 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 heterogeneous graphs. As for communication efficiency, we report the amount of total transmitted parameters of two FL frameworks, *i.e.*, FedAvg and FedDA. 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.

B. Experimental results and analysis



(a) DBLP (b) PubMed

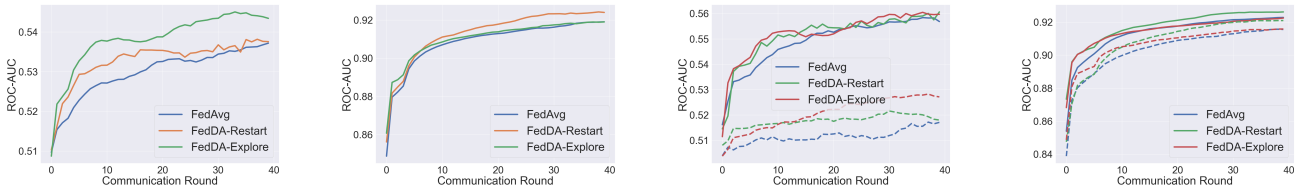
Fig. 5. Average total amount of transmitted gradients on DBLP and PubMed with varying numbers of clients (M). We omit FedProx for simplicity since it has the same amount of transmitted gradients with FedAvg.

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

TABLE II

LINK PREDICTION RESULTS ON FOUR HETEROGENEOUSLY DISTRIBUTED DATASETS WITH 16 CLIENTS. BESIDES AVERAGED ACCURACY, WE ALSO PROVIDE THE CORRESPONDING STD. FEDDA [1]: FEDDA WITH *Restart* STRATEGY, FEDDA [2]: FEDDA WITH *Explore* STRATEGY.

Framework	Model	DBLP		Amazon		LastFM		PubMed	
		ROC-ACU	MRR	ROC-ACU	MRR	ROC-ACU	MRR	ROC-ACU	MRR
Global	S-HGN	0.7750 ± 0.0087	0.9015 ± 0.0045	0.9215 ± 0.0046	0.9604 ± 0.0035	0.8152 ± 0.0042	0.9287 ± 0.0070	0.7670 ± 0.0051	0.8599 ± 0.0036
	D-HGN	0.7810 ± 0.0084	0.9065 ± 0.0048	0.9230 ± 0.0046	0.9647 ± 0.0068	0.8147 ± 0.0024	0.9366 ± 0.0043	0.7719 ± 0.0035	0.8722 ± 0.0022
Local	S-HGN	0.4980 ± 0.0062	0.7503 ± 0.0045	0.7522 ± 0.0945	0.9066 ± 0.0496	0.6128 ± 0.0525	0.7847 ± 0.0309	0.5694 ± 0.0312	0.7760 ± 0.0169
	D-HGN	0.5013 ± 0.0084	0.7582 ± 0.0074	0.7620 ± 0.0948	0.9004 ± 0.0492	0.5964 ± 0.0491	0.7726 ± 0.0331	0.5709 ± 0.0257	0.7794 ± 0.0138
FedAvg	S-HGN	0.5382 ± 0.0074	0.7710 ± 0.0085	0.9187 ± 0.0033	0.9655 ± 0.0029	0.8017 ± 0.0025	0.9233 ± 0.0019	0.6921 ± 0.0118	0.8421 ± 0.0027
	D-HGN	0.5372 ± 0.0060	0.7801 ± 0.0058	0.9203 ± 0.0038	0.9663 ± 0.0049	0.8026 ± 0.0016	0.9227 ± 0.0031	0.6841 ± 0.0101	0.8407 ± 0.0045
FedProx	S-HGN	0.5315 ± 0.0103	0.7507 ± 0.0070	0.9210 ± 0.0006	0.9713 ± 0.0010	0.7802 ± 0.0025	0.9080 ± 0.0030	0.6646 ± 0.0061	0.8310 ± 0.0049
	D-HGN	0.5298 ± 0.0041	0.7523 ± 0.0048	0.9200 ± 0.0013	**0.9725 ± 0.0012**	0.7874 ± 0.0028	0.9145 ± 0.0045	0.6673 ± 0.0093	0.8369 ± 0.0036
FedDA [1]	S-HGN	0.5407 ± 0.0093	0.7784 ± 0.0082	0.9201 ± 0.0051	0.9668 ± 0.0032	0.7964 ± 0.0027	0.9210 ± 0.0020	0.6939 ± 0.0020	0.8459 ± 0.0026
	D-HGN	0.5375 ± 0.0052	0.7787 ± 0.0047	**0.9239 ± 0.0039**	0.9668 ± 0.0038	0.7982 ± 0.0016	0.9230 ± 0.0033	**0.6978 ± 0.0064**	**0.8530 ± 0.0042**
FedDA [2]	S-HGN	0.5422 ± 0.0042	0.7822 ± 0.0070	0.9203 ± 0.0033	0.9635 ± 0.0021	0.8018 ± 0.0030	0.9221 ± 0.0020	0.6938 ± 0.0116	0.8467 ± 0.0028
	D-HGN	**0.5443 ± 0.0058**	**0.7809 ± 0.0065**	0.9190 ± 0.0018	0.9649 ± 0.0025	**0.8114 ± 0.0021**	**0.9371 ± 0.0023**	0.6968 ± 0.0071	0.8456 ± 0.0023



(a) Average performance of D-HGN on DBLP.

(b) Average performance of D-HGN on Amazon.

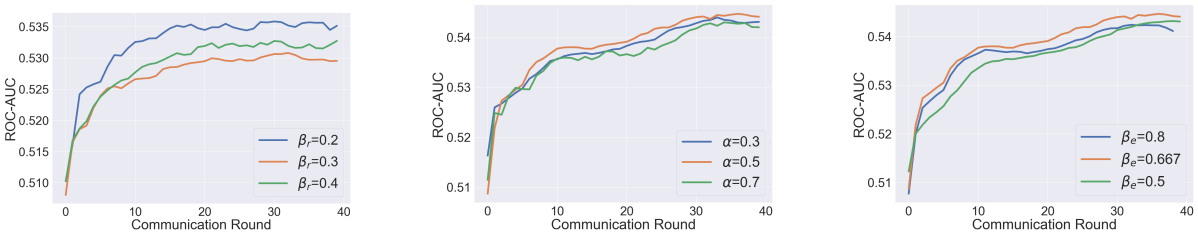
(c) Best and worst performance of D-HGN on DBLP.

(d) Best and worst performance of D-HGN on Amazon.

Fig. 6. Training curves of FedDA compared with FedAvg. In Fig. 6(c) and Fig. 6(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.

FedDA Effectiveness (RQ1): We first compare the performance of our proposed D-HGN with S-HGN on global data in the top part of Tab. II. It appears that D-HGN achieves slightly better performance on both datasets, which indicates the importance of disentangling HGNs towards node and link types even in the centralized training setting. Furthermore, comprehensive experimental results shown in Tab. II indicate that, when we apply these two models under the FedDA framework, due to the ability of type-wise decoupling, D-HGN outperforms S-HGN in most cases. However, this gap does not usually exist under FedAvg, which may indicate that FedAvg is less capable of leveraging more powerful graph models.

As for the comparison between frameworks, 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. II indeed achieve much better performances than vanilla average of local models, which also proves the benefits brought by the collaboration across clients. Furthermore, when comparing FedDA with FedAvg and FedProx, FedDA constantly achieves better performance. Particularly, the *Explore* strategy tends to outperform the others on DBLP and LastFM, while the *Restart* strategy is more effective for Amazon and PubMed. Another thing worth noticing is the gap between FedProx and the other FL frameworks, which could be owing to the graph-specific data heterogeneity. Specifically, the Non-IID nodes and edges in clients are biased not only in feature and label distributions but also in types. While existing methods dealing with Non-IID traditional data can fail to capture the type heterogeneity in our scenario. This further proves the necessity of designing specific FL frameworks for heterogeneous graphs. Moreover, though the final performance enhancement in Tab. II of FedDA compared with FedAvg seems not that significant, FedDA achieves these better performances with much fewer communication rounds, activated clients, transmitted parameters, and a more stable training process, which is the main focus of this work. In other



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

Fig. 7. Training curves of FedDA with different parameters setting. All curves are generated by D-HGN on DBLP with 16 clients.

words, if we have a strict budget for communication costs (e.g. limited communication rounds), FedAvg will perform much worse and FedDA will have more significantly enhanced performance. Specifically, as shown in Fig. 6, if we look at 20 communication rounds instead of 40, we can observe that both of our activation strategies can achieve larger improvements over FedAvg on DBLP and Amazon. We will further discuss such efficiency benefits in RQ2, and the stabilized training process in RQ3.

Communication Efficiency (RQ2): We compare the amount of total transmitted gradients of FedDA with FedAvg in Fig. 5. To be consistent with Tab. II and conduct a fair comparison, we are presenting the results after 40 communication rounds for all frameworks, after which FedAvg is converged to its global optimum. However, FedDA often converges faster than FedAvg, and we will discuss it in the next RQ. Even so, according to Fig. 5, both implementation strategies of FedDA can reduce a significant amount of transmitted parameters, especially for D-HGN, which further proves that the dynamic activation mechanism effectively reduces the communication cost as expected. Note that, the disentangled rate $\frac{N_d}{N}$ for parameters in S-HGN is 0.769 for DBLP and 0.595 for Amazon, while in D-HGN, it is 0.944 and 0.878, respectively. According to our analysis in Sec. V-C3, a model with high $\frac{N_d}{N}$ is potentially more efficient, which is exactly the case in Fig. 5. Besides, although we only compute the amount of transmitted gradients as a measure of efficiency, fewer transmitted gradients are naturally equivalent to less computation on clients (especially the deactivated ones) and the server.

Convergence Studies (RQ3): Taking D-HGN working on DBLP and Amazon datasets with 16 clients as an example, we present the convergence curves in Fig. 6. Notably, $Restart$ and $Explore$ can both achieve better performance with significantly fewer communication rounds. For instance, FedDA with $Explore$ on DBLP can achieve a score over 0.535 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 much better performance. Moreover, if we take the number of necessary communication rounds into consideration while calculating the transmitted cost, the efficiency enhancement would become even more significant. For instance, if we run FedDA with $Explore$ strategy for only 20 rounds on DBLP, we will have a model just as effective as FedAvg with 40 rounds, saving approximately 75% transmitted parameters. Furthermore, as shown in Fig. 6(c) and Fig. 6(d), if we consider the best and the worst performance of these frameworks like in Sec. IV, it appears that FedDA could also narrow the gap between maximum and minimum even compared with FedAvg with $C = 1$, which is a significant improvement given the results in Fig. 2(a) and Fig. 2(b). It further proves that the dynamic activation strategy is capable of stabilizing the FL process.

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. 7. In Fig. 7(a), we fix other parameters and study β_r for $Restart$ strategy, while for Fig. 7(b) and Fig. 7(c) we present $Explore$ strategy with different α and β_e. All the results are generated by training D-HGN on DBLP with 16 clients.

Observed from Fig. 7(a), β_r affects the final performance significantly. Thus, choosing a proper β_r, which controls the number of communication rounds before re-initializing, is crucial to elevate the final testing accuracy. Furthermore, if the total number of communication rounds is fixed, then a smaller β_r will tolerate fewer clients being activated before reactivating them all, leading to better communication efficiency. As for α in Fig. 7(b), apparently, it has a negligible effect on the final global performance, but a too large α can lead to an unstable training process. We infer that such observations can be attributed to the too-aggressive client deactivation threshold, which indicates that the server will repeatedly activate clients with rather trivial contributions. At last, for β_e, though the analysis in Sec. V-C3 shows a positive relationship between the β_e and the amount of transmitted parameters, as we only consider efficient models with the best performance, in this work, we settle with $\beta_e = 0.667$ since it brings the most effective model according to Fig. 7(c).

VII. Conclusion

In this work, we study the demanded yet challenging setting of FL across distributed Non-IID heterogeneous graphs. We revisit the vanilla FedAvg protocol over heterogeneous graphs and surprisingly discover that the partial activation of clients and parameters can potentially benefit the final model's performance and communication efficiency. 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. Both theoretical and 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.

APPENDIX

In appendix, we provide the formal proof of theorems presented in our main paper.

A. Proof of Theorem 1

Proof. According to our assumption, we know that

$$g = \nabla \mathcal{L}(\mathbf{w}),\ g' = \nabla \mathcal{L}(\mathbf{w}') \tag{13}$$

Since $\mathbf{w}' = \mathbf{w} - \eta \cdot g$, by the Lipschitz assumption, we have

$$\begin{aligned} \|g' - g\|_2 &= \|\nabla \mathcal{L}(\mathbf{w}') - \nabla \mathcal{L}(\mathbf{w})\|_2 \\ &\leq L_g \cdot \|\mathbf{w}' - \mathbf{w}\|_2 = L_g \cdot \eta \cdot \|g\|_2 \leq L_g \cdot L_f \cdot \eta \end{aligned} \tag{14}$$

where the last inequality holds because the ℓ_2-norm of a function's gradients is always upper bounded by the function's Lipschitz continuous constant, and one can easily prove it by mean value theorem.

Only requiring $\|g' - g\|_2$ to be small is not enough to guarantee the inequality we want to prove. The main idea for the rest of the proof is to find the maximal amount of shift of each g_i and the average $\bar{g}$, such that the relative position of $\bar{g}$ and the g_i closest to $\bar{g}$ do NOT change after one step of gradient descent. Denote the two gradients that are closest to $\bar{g}$ as g_j and g_{j+1}. Given $\|g' - g\|_2 \leq L_g \cdot L_f \cdot \eta$, we know that

$$\|g' - g\|_1 \leq \sqrt{|\mathcal{I}|} \cdot \|g' - g\|_2 \tag{15}$$

where $\sqrt{|\mathcal{I}|}$ is essentially the dimension of g and g'. Hence,

$$\left(\sqrt{|\mathcal{I}|} + \sqrt{|\mathcal{I}|^{-1}}\right) \cdot \|g' - g\|_2 < \tau\epsilon \tag{16}$$

Plugging Eq. (14) into above, we have

$$\eta < \left(\left(\sqrt{|\mathcal{I}|} + \sqrt{|\mathcal{I}|^{-1}}\right) L_f L_g\right)^{-1} \cdot \tau\epsilon \tag{17}$$

□

B. Proof of Theorem 2 and 3

Proof of Theorem 2. First, notice that FedDA with *Restart* strategy can be regarded as a generalization of *FedAvg* with full device participation as defined in [43]. Formally speaking, suppose restarting strategy is applied every R iterations, then for iterations $1, 1+R, 1+2R, \cdots, 1+ZR$, $Z \in \mathbb{N}^+$, all devices are employed, while for the rest of iterations partial devices are employed due to the dynamic activation mechanism. If we only consider FedDA with iterations where all devices participate, the proof directly follows that in Theorem 1 in [43], only differing in the number of iterations that T should be replaced with T/R. Hence, the rest of the proof is to show that for iterations with partial device participation, the loss function does NOT increase, which combined with Theorem 1 in [43] will be sufficient to guarantee that FedDA with restarting strategy can converge.

By definition, we have $\mathcal{L} = \sum_{k=1}^{N} p_k \mathcal{L}_k$, where $\mathcal{L}_k$, p_k is local loss, device aggregation weight, respectively. For some iteration t that only K devices participate ($K < N$). Without loss of generality, we assume the first K devices participate in iteration t. By definition of gradient descent,

$$\forall\ k \leq K, \quad \mathcal{L}_k(\mathbf{w}_k^{(t)}) - \mathcal{L}_k(\mathbf{w}_k^{(t-1)}) < 0 \tag{18}$$

where $\mathbf{w}_k^{(t)} = \mathbf{w}_k^{(t-1)} - \eta_k g_k$, where η_k is the learning rate for device k. For those devices that do NOT participate in iteration t, we have

$$\forall\ k > K, \quad \mathcal{L}_k(\mathbf{w}_k^{(t)}) - \mathcal{L}_k(\mathbf{w}_k^{(t-1)}) = 0 \tag{19}$$

since their model parameters do not change. Combining both equations above finishes the proof.

□

Proof of Theorem 3. The general idea here is to show that our FedDA with *explore* strategy is a special case of *FedAvg* with partial device participation as defined in Theorem 2 and 3 in [43]. The core part to prove lies in showing that the *explore* strategy gives an unbiased estimator. Due to the limited space here, we omit some unnecessary details and please refer to Sec. B.2 in [43] if interested.

The key idea is to regard FedDA with *explore* strategy as stratified sampling scheme. To see this, suppose given two deterministic sets $\mathcal{X} = \{x_1, x_2, \cdots, x_{N_1}\}$ and $\mathcal{Y} = \{y_1, y_2, \cdots, y_{N_2}\}$, and define $\mathcal{Z} = \mathcal{X} \cup \mathcal{Y}$. Define S_K as a set of random samples by stratified sampling without replacement from $\mathcal{Z}$, as $S_K = \{x_1, x_2, \cdots, x_{K_1}, y_1, y_2, \cdots, y_{K_2}\}$ with $K = K_1 + K_2$. Here we consider stratified sampling with uniform distribution within each strata. Then,

$$\begin{aligned} \mathbb{E}[\sum S_K] &= \mathbb{E}[\sum S_{\mathcal{X}}] + \mathbb{E}[\sum S_{\mathcal{Y}}] \\ &= K_1 \cdot \sum_{i=1}^{N_1} \frac{1}{N_1} x_i + K_2 \cdot \sum_{i=1}^{N_2} \frac{1}{N_2} y_i \\ &= \frac{K_1}{N_1} \sum_{i=1}^{N_1} x_i + \frac{K_2}{N_2} \sum_{i=1}^{N_2} y_i \\ &\coloneqq K_1 \cdot \bar{\mathcal{X}} + K_2 \cdot \bar{\mathcal{Y}} \end{aligned} \tag{20}$$

where $\bar{\mathcal{X}}$ and $\bar{\mathcal{Y}}$ is defined as average over all elements in set $\mathcal{X}$ and $\mathcal{Y}$, respectively. On the other hand,

$$\bar{\mathcal{Z}} = \frac{\sum \mathcal{X} + \sum \mathcal{Y}}{N_1 + N_2} = \frac{N_1}{N_1 + N_2} \cdot \bar{\mathcal{X}} + \frac{N_2}{N_1 + N_2} \cdot \bar{\mathcal{Y}} \tag{21}$$

Suppose $N_1 = cN_2$, $c \in \mathbb{R}^+$. If we choose $K_1 = cK_2$, then it directly follows that there exists some constant $C \in \mathbb{R}^+$, such that

$$\mathbb{E}[\sum S_K] = C \cdot \bar{\mathcal{Z}} \tag{22}$$

which proves that our sampling scheme S_K is unbiased over $\bar{\mathcal{Z}}$ up to a scaling constant. Hence, our sampling scheme can be regarded as a special case of scheme 2 in Sec. B.2 in [43] and the rest of the proof follows there directly.

In practice, set $\mathcal{Y}$ corresponds to the devices that 100% participate, so K_2 will be set to N_2. On the other hand, set $\mathcal{X}$ represents the devices that are first de-activated and then we uniformally sample some from $\mathcal{X}$ following the *explore* strategy. We can control the ratio c to make sure our sampled estimator is unbiased.

□

ACKNOWLEDGE

This research was partially supported by the internal funds and GPU servers provided by the Computer Science Department of Emory University, as well as the collaborative Research Grant funding provided by the Halle Institute for Global Research at Emory University and the Office of International Cooperation and Exchanges at Nanjing University.

REFERENCES

[1] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han, "Heterogeneous network representation learning: A unified framework with survey and benchmark," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2020.

[2] C. Yang, Y. Feng, P. Li, Y. Shi, and J. Han, "Meta-graph based hin spectral embedding: Methods, analyses and insights," in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2018.

[3] C. Yang, J. Zhang, and J. Han, "Neural embedding propagation on heterogeneous networks," in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2019.

[4] C. Yang and J. Han, "Revisiting citation prediction with cluster-aware text-enhanced heterogeneous graph neural networks," in *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, 2023.

[5] E. Sügis, J. Dauvillier, A. Leontjeva, P. Adler, V. Hindie, T. Moncion, V. Collura, R. Daudin, Y. Loe-Mie, Y. Herault *et al.*, "Hena, heterogeneous network-based data set for alzheimer's disease," *Scientific data*, vol. 6, pp. 1–18, 2019.

[6] Z. Wang, T. Xia, R. Jiang, X. Liu, K.-S. Kim, X. Song, and R. Shibasaki, "Forecasting ambulance demand with profiled human mobility via heterogeneous multi-graph neural networks," in *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, 2021.

[7] C. Yang, M. Liu, F. He, X. Zhang, J. Peng, and J. Han, "Similarity modeling on heterogeneous networks via automatic path discovery," in *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2018.

[8] C. Chen, W. Ma, M. Zhang, Z. Wang, X. He, C. Wang, Y. Liu, and S. Ma, "Graph heterogeneous multi-relational recommendation," in *TNNLS*, 2021.

[9] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *TNNLS*, 2021.

[10] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," 2021.

[11] X. Zhu, J. Wang, Z. Hong, and J. Xiao, "Empirical studies of institutional federated learning for natural language processing," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2020.

[12] K. Zhang, C. Yang, X. Li, L. Sun, and S. M. Yiu, "Subgraph federated learning with missing neighbor generation," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[13] H. Xie, J. Ma, L. Xiong, and C. Yang, "Federated graph classification over non-iid graphs," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[14] C. He, K. Balasubramanian, E. Ceyani, C. Yang, H. Xie, L. Sun, L. He, L. Yang, P. S. Yu, Y. Rong, P. Zhao, J. Huang, M. Annavaram, and S. Avestimehr, "Fedgraphnn: A federated learning system and benchmark for graph neural networks," 2021.

[15] C. Chen, W. Hu, Z. Xu, and Z. Zheng, "Fedgl: Federated graph learning framework with global self-supervision," 2021.

[16] F. Chen, P. Li, T. Miyazaki, and C. Wu, "Fedgraph: Federated graph learning with intelligent sampling," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2022.

[17] Q. Lv, M. Ding, Q. Liu, Y. Chen, W. Feng, S. He, C. Zhou, J. Jiang, Y. Dong, and J. Tang, "Are we really making much progress? revisiting, benchmarking, and refining heterogeneous graph neural networks," *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2021.

[18] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[19] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, pp. 1–207, 2019.

[20] J. Liu, J. Huang, Y. Zhou, X. Li, S. Ji, H. Xiong, and D. Dou, "From distributed machine learning to federated learning: A survey," *Knowledge and Information Systems*, pp. 1–33, 2022.

[21] J. Zhou, C. Chen, L. Zheng, H. Wu, J. Wu, X. Zheng, B. Wu, Z. Liu, and L. Wang, "Vertically federated graph neural network for privacy-preserving node classification," 2021.

[22] G. Mei, Z. Guo, S. Liu, and L. Pan, "Sgnn: A graph neural network based federated learning approach by hiding structure," in *Proceedings of the IEEE International Conference on Big Data (ICBD)*, 2019.

[23] C. Y. Han Xie, Li Xiong, "Federated node classification over graphs with latent link-type heterogeneity," in *Proceedings of the Workshop on Federated Learning over Graph Data: The ACM International Conference on Information and Knowledge Management (CIKM-FedGraph)*, 2022.

[24] C. Wu, F. Wu, Y. Cao, Y. Huang, and X. Xie, "Fedgnn: Federated graph neural network for privacy-preserving recommendation," 2021.

[25] M. Chen, W. Zhang, Z. Yuan, Y. Jia, and H. Chen, "Fede: Embedding knowledge graphs in federated setting," in *Proceedings of the International Joint Conference on Knowledge Graphs (IJCKG)*, 2021, pp. 80–88.

[26] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 29, pp. 17–37, 2016.

[27] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2020.

[28] B. Wang, A. Li, H. Li, and Y. Chen, "Graphfl: A federated learning framework for semi-supervised node classification on graphs," 2020.

[29] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, pp. 1–54, 2019.

[30] T. Zhao, X. Zhang, and S. Wang, "Graphsmote: Imbalanced node classification on graphs with graph neural networks," in *Proceedings of the International Web Search and Data Mining Conference (WSDM)*, 2021.

[31] S. Saxena, O. Tuzel, and D. DeCoste, "Data parameters: A new family of parameters for learning a differentiable curriculum," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[32] X. Wang, Y. Chen, and W. Zhu, "A survey on curriculum learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[33] M. Sachan and E. Xing, "Easy questions first? a case study on curriculum learning for question answering," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

[34] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, "Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[35] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

[36] Y. Wang, W. Wang, Y. Liang, Y. Cai, and B. Hooi, "Curgraph: Curriculum learning for graph classification," in *Proceedings of the International World Wide Web Conference (WWW)*, 2021.

[37] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2013.

[38] T. Huang, Y. Dong, M. Ding, Z. Yang, W. Feng, X. Wang, and J. Tang, "Mixgcf: An improved training method for graph neural network-based recommender systems," in *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2021.

[39] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-iid data quagmire of decentralized machine learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[40] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han, "Heterogeneous network representation learning: A unified framework with survey and benchmark," *TKDE*, 2020.

[41] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *Proceedings of the International Conference on Learning Representations, International Conference on Learning Representations (ICLR)*, 2018.

[42] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[43] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[44] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang, "Representation learning for attributed multiplex heterogeneous network," *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2019.

[45] B. Plank and R. van Dalen, "Citetracked: A longitudinal dataset of peer reviews and citations." in *Proceedings of the Joint Workshops on Bibliometric-enhanced Information Retrieval and Natural Language Processing for Digital Libraries (BIRNDL)*, 2019.

[46] D. Cummings and M. Nassar, "Structured citation trend prediction using graph neural networks," in *Proceedings of the International Conference on Acoustics, Speech, & Signal Processing (ICASSP)*, 2020.

[47] S. Jiang, B. Koch, and Y. Sun, "Hints: Citation time series prediction for new publications via dynamic heterogeneous information network embedding," in *Proceedings of the International World Wide Web Conference (WWW)*, 2021.

[48] I. Cantador, P. Brusilovsky, and T. Kuflik, "Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011)," in *Proceedings of the Fifth ACM Conference on Recommender Systems*, ser. RecSys '11. Association for Computing Machinery, 2011.