Deep Efficient Private Neighbor Generation for Subgraph Federated Learning

Ke Zhang[§] *ClusterTech Limited* HKSAR, China
Lichao Sun *Lehigh University* Bethlehem, U.S.A.
Bolin Ding *Alibaba Group* Hangzhou, China
Siu Ming Yiu *The University of Hong Kong* HKSAR, China
Carl Yang[*] *Emory University* Atlanta, U.S.A.

***Abstract*—Behemoth graphs are often fragmented and separately stored by multiple data owners as distributed subgraphs in many realistic applications. Without harming data privacy, it is natural to consider the *subgraph federated learning* (subgraph FL) scenario, where each local client holds a subgraph of the entire global graph, to obtain globally generalized graph mining models. To overcome the unique challenge of incomplete information propagation on local subgraphs due to missing cross-subgraph neighbors, previous works resort to the augmentation of local neighborhoods through the joint FL of missing neighbor generators and GNNs. Yet their technical designs have profound limitations regarding the utility, efficiency, and privacy goals of FL. In this work, we propose FedDEP to comprehensively tackle these challenges in subgraph FL. FedDEP consists of a series of novel technical designs: (1) Deep neighbor generation through leveraging the GNN embeddings of potential missing neighbors; (2) Efficient pseudo-FL for neighbor generation through embedding prototyping; and (3) Privacy protection through noiseless edge-local-differential-privacy. We analyze the correctness and efficiency of FedDEP, and provide theoretical guarantees on its privacy. Empirical results on four real-world datasets demonstrate the clear benefits of our proposed techniques.**

***Index Terms*—Federated Learning, Graph Neural Networks**

I. Introduction

Graph data mining, one of the most important research domains for knowledge discovery, has been revolutionized by Graph Neural Networks (GNNs), which have established state-of-the-art performance in various domains such as social platforms [1], e-commerce [2], transportation [3], bioinformatics [4], and healthcare [5]. The power of GNNs benefits from training on real-world graphs with millions to billions of nodes and links [6], [7]. Nowadays, emerging graph data from many realistic applications, such as recommendation, drug discovery, and infectious disease surveillance, are naturally fragmented, forming distributed graphs of multiple "data silos". Moreover, due to the increasing concerns about data privacy and regulatory restrictions, directly transferring and sharing local data to construct the entire global graph for GNN training is unrealistic [8], [9].

Federated learning (FL) is a promising paradigm for distributed machine learning that addresses the data isolation problem, which has recently received increasing attention in various applications [10], [11]. Despite its successful applications in domains like computer vision [12] and natural language processing [13] where data samples (i.e., images or documents) hardly interact with each other, FL over graph data manifests unique challenges due to the complex node dependencies, structural patterns, and feature-link correlations. In this work, we focus on one of the most common and challenging scenarios of *federated learning over distributed subgraphs* (subgraph FL), where clients hold subgraphs of largely disjoint sets of nodes and their respective links, as illustrated in Fig. 1 (b) and (c). One unique challenge in subgraph FL is the incomplete neighborhood of nodes in the local subgraphs caused by cross-subgraph missing neighbors, that is, potential neighboring nodes captured by other local subgraphs. This phenomenon cannot be properly handled by applying generic FL mechanisms such as FedAvg for GNN training. Targeting this limitation, Zhang et al. propose FedSage [14], where a *missing neighbor generator* is collaboratively learned across clients to retrieve cross-subgraph missing neighbors and better approximate GNN training on the entire global graph (Fig. 1 (d)). The success of FedSage justifies the necessity of obtaining complete information in local neighborhoods. However, the designs of FedSage have several deficiencies in complex and sensitive real-world scenarios, regarding *utility*, *efficiency*, and *privacy*.

Limited Utility. The missing neighbor generator in FedSage can only recover 1-hop missing neighbors and does not further propagate their features to other neighboring nodes. As illustrated in Figure 1 (e), for predictions where neighbors further than 1 hop away are important, the model will still fail. Such limitation can also be verified by the limited performance gain of FedSage+ over vanilla FedSage without neighbor generator in [14] as well as our experimental results in Table III.

Significant Overhead. The FL training of missing neighbor generators in FedSage incurs substantial inter-client communication costs (Step I in Fig. 1 (d)), in addition to the standard client-server communication in FL (Step III), and heavy intra-client computations (Step II). Specifically, for each FL iteration, each client needs to broadcast the generated node embeddings to all other clients and receive training gradients for the neighbor generator in Step I, and each client needs to repeatedly search across its entire node set to find the most likely cross-subgraph missing neighbors in Step III.

Privacy Concerns. For subgraph FL, FedSage shares GNN gradients and node embeddings instead of raw data, but without specific privacy protection, the gradients and embeddings

[§] *This work was done while at the University of Hong Kong.*
[*] *Corresponding author (j.carlyang@emory.edu).*

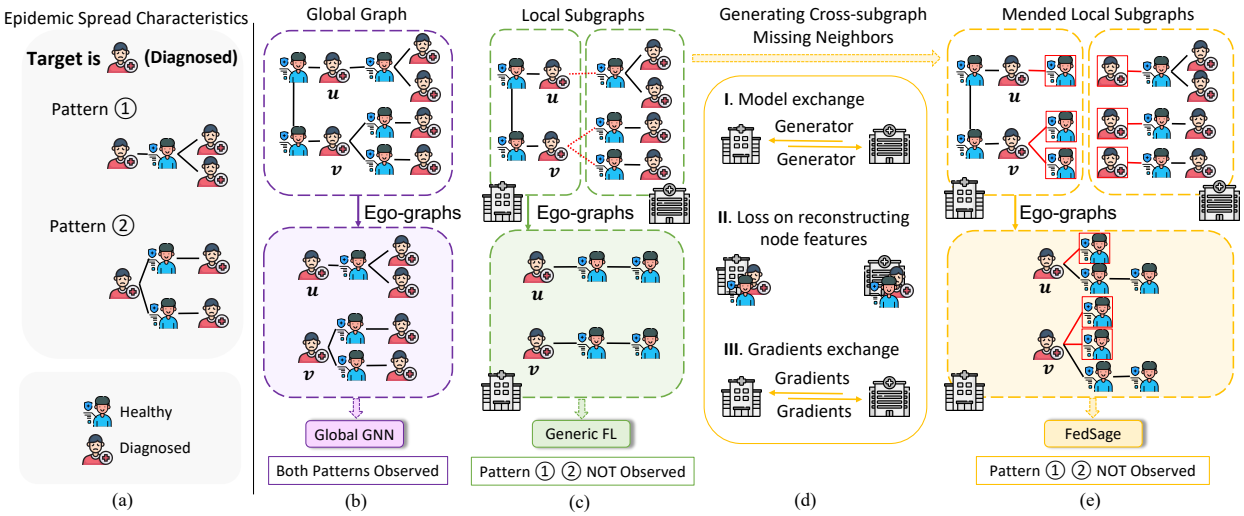

Fig. 1: A toy example of modeling the spread of infectious disease in a distributed subgraph FL system. The black lines are the close contact relations between people, and the dashed red lines are the cross-subgraph missing links. Red solid lines are the generated links, and the people figures with red solid rectangles are the generated neighbors. (a) The reason for a target to be diagnosed when his/her direct contacts are all healthy can be attributed to Pattern 1 : some of the healthy neighbors have direct contacts with many diagnosed ones, or Pattern 2 : many of those healthy neighbors have direct contacts with diagnosed ones. (b) If the global graph is available, both patterns are observable and centralized GNN can correctly identify the reasons for both u and v to be infected. (c) In the more realistic setting of local subgraphs, neither of the patterns is observable and GNN obtained from generic FL (such as FedAvg) will fail to learn why u and v are infected. (d) FedSage tries to achieve the recovery of 1-hop missing neighbors across local subgraphs through three steps, which require significant extra communication and computation. (e) Unfortunately, even if all 1-hop missing neighbors can be generated accurately, GNN obtained through FedSage will still fail because the correct patterns require access to deeper missing neighbors.

directly computed from raw data are prone to privacy leakage, such as via inference attacks [15], [16] and reconstruction attacks [17], [18]. Unlike FedSage, FedGNN [19] provides privacy protections by leveraging homomorphic encryption [20] and DPSGD techniques [21]. However, it relies on an additional trusted authority to achieve strict privacy guarantees, which is not available in general subgraph FL.

Herein, we propose Subgraph Federated Learning with Deep Efficient Private Neighbor Generation (FedDEP) to address the unique utility, efficiency, and privacy challenges in the important and realistic setting of subgraph FL.

Utility-wise: Deep Neighbor Generation and Embedding-fused Graph Convolution (DGen). To enhance the modeling of cross-subgraph missing neighbors in the system without exponentially increasing computation and communication overheads, we propose a deep neighbor generator, DGen. It generates missing neighbors in depth, by leveraging the GNN embeddings of generated neighbors. The generated embeddings contain information from the target node's multiple hops of neighbors [22], [23] and include richer context beyond single node features generated in FedSage. To incorporate the generated deep neighbors into GNN training, we propose a novel embedding-fused graph convolution process for the system to obtain the global classifier.

Efficiency-wise: Deep Embedding Prototyping and Pseudo-FL (Proto). To reduce the intra-client computation, we cluster the GNN embeddings of nodes in each client to construct sets of missing neighbor prototypes. Instead of repeated exhaustive searches for closest neighbors across a client's entire node set as in FedSage, we can find the closest prototypes across the much smaller prototype sets. To further reduce the inter-client communication, we propose pseudo-FL by sharing the prototype embeddings across the system before the training of DGen. Thus, clients can conduct closest neighbor searches locally without communications while still achieving FL for DGen. Similarly to [24], sharing the prototype embeddings instead of node embeddings can also lead to empirical privacy benefits due to the difficulty in inference attacks from aggregated models.

Privacy-wise: Noise-free differential privacy through random sampling (NFDP). We aim to theoretically guarantee rigorous edge-local-differential-privacy (edge-LDP), which protects edges' existence within local node's neighborhoods in distributed subgraphs [25]. Particularly, we achieve noise-free edge-LDP by transferring noise-free differential privacy from general domains [26] to edge-LDP, without embracing complicated cryptology techniques, explicitly perturbing shared data, or introducing additional roles into the system as previous work [19]. Technically, we incorporate two stages of random sampling into FedDEP, *i.e.*, (1) mini-batching: random neighborhood sampling in each graph convolution layer [27]; and (2) Bernoulli-based generation selection: randomly sampling

generated deep neighbors by a Bernoulli sampler in DGen.

Extensive experiments on four real-world graph datasets justify the utility, efficiency, and privacy benefits of FedDEP.

II. Related Works

A. Federated Learning for Graphs

With massive graph data separately stored by distributed data owners, recent research has emerged in the field of FL over graph data. Some studies propose FL methods for tasks on distributed knowledge graphs, such as recommendation or representation learning [28]–[31]. Another direction is for the scenarios where every client holds a set of small graphs, such as molecular graphs for drug discovery [32]. In this work, we consider subgraph FL, where each client holds a subgraph of the entire global graph, and the only central server is dataless. The instrumental isolation of data samples leads to incomplete structural features of local nodes due to cross-subgraph neighbors missing not at random, which is fundamentally different from the centralized graph learning scenarios with unbiased sparse links [33] or randomized DropEdge [34].

To deal with the missing neighbor problem in subgraph FL, existing works [14], [19], [35], [36] propose to augment local subgraphs by retrieving missing neighbors across clients, and then mend the subgraphs with the retrieved neighbors. FedGraph [36] considers a relaxed scenario where the existences of inter-subgraph neighbors are known for corresponding clients. Moreover, it requests the central server to manage the FL process with auxiliary data. FedSage [14] primarily focuses on the design of the missing neighbor generator without considering the important aspects of efficiency and privacy. FedHG [35] studies the heterogeneous subgraph FL systems where graphs consist of multiple types of nodes and links, and it only protects the partial privacy of certain types of nodes in the system. FedGNN [19] equips its augmentation with privacy guarantees based on an additional trusted authority.

None of them provides a complete solution to the utility, efficiency, and privacy of subgraph FL.

B. Privacy-Preserving Learning for Graphs

Privacy-preserving learning over graph data has been widely studied. Differential Privacy (DP) [37] is a widely applied privacy concept in this field, which describes the privacy of a method in protecting individual samples while preserving the analytical properties of the entire dataset. A prevalent approach in attaining a graph mining model with general DP is DPSGD [21], which injects designed noise into clipped gradients during model training. For centralized training scenarios, DPGGAN [38] incorporates DPSGD to achieve DP for individual links on original graphs. In FL systems, VFGNN [39] and FedGNN [19] leverage DPSGD and cryptology techniques to obtain rigorous privacy guarantees for federated graph learning. Meanwhile, to achieve general DP on graphs, there are some other noised-injecting based methods. Previous works of centralized learning [40]–[42], and FKGE [28] for FL systems, guarantee their proposed techniques with general DP by applying noise perturbation.

However, general DP does not depict the privacy guarantees for sensitive node features, edges, or neighborhoods, on distributed graphs. Edge local DP and node local DP (edge-LDP and node-LDP) are two specific types of DP targeting local nodes' neighbor lists [25]. These novel DP definitions better fit the graph learning that learns from multiple neighbor lists, and match the privacy goal of protecting nodes' local neighborhoods.

As illustrated in Definition 2.2 in [25], edge-LDP defines how much a model tells for two neighborhoods that differ by one edge, while node-LDP promises a model's max leakage for all possible neighborhoods. In contrast to node-LDP, which is much stronger and can severely hinder the graph model's utility, edge-LDP precisely illustrates the local DP for local neighborhoods without overly constraining the model.

There are several works analyzing edge-LDP over distributed graph data. Qin et al. [25] propose a decentralized social graphs generation technique with the edge-LDP. Imola et al. [43] analyze the edge-LDP of the proposed shuffle techniques in handling the triangle and 4-cycle counting for neighbor lists of distributed users. Lin et al. [44] propose Solitude, an edge-LDP collaborative training framework for distributed graphs, where each client shares its perturbed local graph for the training. However, different from our subgraph FL setting, its central server (data curator) has access to node identities and labels. To the best of our knowledge, we are the first to leverage edge-LDP in the FL setting.

III. Background

A. Problem Formulation

a) ***Distributed Subgraph System****:* We denote a global graph as $G = \{V, E, X\}$, where V is the node set, X is the respective node feature set, and E is the edge set. In the subgraph FL system, we have one central server S and M clients with distributed subgraphs. $G_i = \{V_i, E_i, X_i\}$ is the subgraph owned by D_i, for $i \in [M]$, where $V = \bigcup_{i=1}^{M} V_i$. In this system, we assume that the dataless central server S only maintains a graph model, and no direct sharing of nodes nor edges is allowed. For simplicity, we assume no overlapping nodes shared across data owners, *i.e.*, $V_i \setminus V_j = \emptyset$ for any $i \neq j \in [M]$. Thus, for an edge $e_{v,u} \in E$, where $v \in V_i$ and $u \in V_j$, we have $e_{v,u} \notin E_i \cup E_j$. That is, $e_{v,u}$ might exist in reality but is missing from the whole system.

b) ***Goal****:* The system exploits an FL framework to collaboratively learn a global node classifier F on isolated subgraphs in clients, without raw graph data sharing. The learnable weights W in F are optimized following the distribution of the global graph G. We formalize the problem as finding W that minimizes the risk

$$W^* = \arg\min_W R(F(W)) = \arg\min_W \frac{1}{M} \sum_i^M R_i(F_i(W))),$$

where $R_i(F_i(W)) := \mathbb{E}[\ell(F_i(W; G_i), Y_i))]$ is the local empirical risk, and $\ell(\cdot)$ is a task-specific loss function.

B. Preliminaries

In this subsection, we revisit the popular existing subgraph federated learning framework, i.e., FedSage+, the variant of FedSage with the proposed missing neighbor generator (NeighGen) [14]. For simplicity, in this paper, we refer to this stronger variant as FedSage.

1) Neighbor Generation: The proposed NeighGen in [14] includes an encoder H^e and a generator H^g. For a node v on G_i, NeighGen generates its missing neighbors by taking in its K-hop ego-graph $G_i^K(v)$. Specifically, it predicts the number of v's missing neighbors n_v , and predicts the respective feature set x_v.

2) Cross-subgraph Neighbor Reconstruction: To obtain ground truth for supervising NeighGen without actually seeing the missing neighbors, each client simulates the missing neighbor situation by randomly holding out a pre-determined portion of the nodes and all links involving them. To allow a NeighGen model to generate diverse and realistic missing neighbors, the system conducts federated cross-subgraph training as follows.

1) Each client D_i sends its local NeighGen's generator H^g and its input to all other clients D_j.
2) D_j computes the cross-subgraph feature reconstruction loss $L_{i;j}^f$ between real node features on G_j and the generated ones from received data.
3) D_j sends $L_{i;j}^f$'s gradients back to D_i via server S.
4) D_i computes the total gradients of cross-subgraph neighbor reconstruction loss $L_i^f = {}^n\sum_{j2[M]} L_{i;j}^f$ by summing up all received gradients from other clients. Notably, $L_{i;i}^f$ is the local neighbor reconstruction loss computed on local ground truth obtained from hidden nodes and edges.

To attain the generalized final classifier, in FedSage, data owners federally train a shared model of NeighGen with a GraphSage classifier, where the classifier learns on nodes drawn from local subgraphs mended with the generated neighbors. For more technical details of the process and equations, please refer to the original paper of FedSage [14].

IV. FedDEP

A. Utility Elevation through DGen

The demanding utility challenge left by FedSage is how to further enrich local node contexts regarding deeper missing neighbors. However, directly generating deeper neighbors will incur exponential increments for intra-client computation and inter-client communication. To deal with this, we propose to leverage the GNN encoder and generate deep embeddings of missing neighbors that can capture their multi-hop local contexts in the corresponding subgraphs. Fig. 2 illustrates this technical motivation, regarding the toy example in Fig. 1.

1) Deep Neighbor Generation: Inspired by the design of NeighGen in FedSage, we propose a deep neighbor generator DGen, whose architecture is shown in the middle of Fig. 3. ^{e} and $f\ ^d;\ ^f g$ are the learnable parameters of DGen's two components, *i.e.*, the GNN encoder H^e and the embedding generator H^g, respectively.

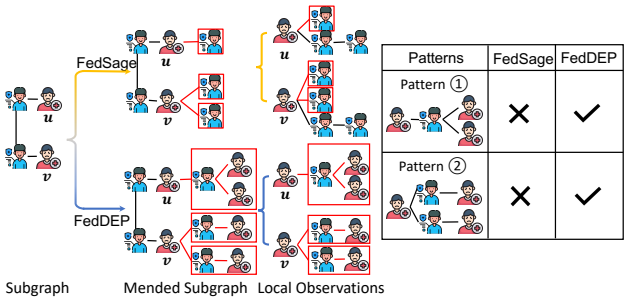

Fig. 2: Technical motivation of FedDEP against FedSage. FedDEP generates information of multiple hops of neighbors to provide the subgraph with richer information for local nodes, compared to the missing neighbor features generated by FedSage. Hence, FedDEP assists the local subgraph to correctly capture both patterns.

Unlike NeighGen which generates node features of missing neighbors, DGen generates node embeddings instead. Particularly, for a node v on graph G_i, we have its generated missing deep neighbors as

$$\begin{aligned} n_v &= (\ ^{d\top}\ H^e(G_i^K(v);\ ^e)); \\ z_v &= Ber_r\ \ ^{f\top}\ [H^e(G_i^K(v);\ ^e) + \mathrm{N}(0;1)]\ ;n_v\ ; \end{aligned} \tag{1}$$

where n_v 2 N, z_v 2 $\mathrm{R}^{n_v\ \ d_z}$, and d_z is the dimension of node embeddings. $Ber_r(a;b)$ is a Bernoulli sampler that independently samples b records from a following $Ber(r)$, with r as a constant. In this process, based on the original neighborhood of v, DGen first predicts the number of its missing neighbors n_v and then samples n_v embedding vectors for them. Sampling allows the whole process to be generative and trained through variational inference for enhanced robustness [45].

2) Embedding-fused Graph Convolution: After mending the local subgraph with generated deep neighbors, *i.e.*, attaching the 1-hop neighbors with deep embeddings, every client obtains its mended local subgraph $G_i = fV_i; E_i; X_i; Z_i g$. Recall that our ultimate goal is to obtain a node classifier F for node classification tasks. Obviously, existing GNNs with vanilla graph convolution process is incapable to incorporate the mended nodes with deep embeddings as node features, due to the difference between their feature spaces. To properly conduct node classification on $\bar{G}_i$, we propose the embedding-fused graph convolution mechanism.

*a) **Downstream Classifier** F:* We specify the downstream classifier F as a model of K-layer embedding-fused graph convolution. For a queried node v 2 V, F integrates nodes together with all mended embedding sets on v's K-hop ego-graph $G_i^K(v)$ from $\bar{G}_i$. The entire convolution is achieved by learnable weights $W = fW^{(k)} jk\ \ 2\ \ [0;K]g$, where $W^{(0)}$ 2 $\mathrm{R}^{d_h\ \ (d_x+d_z)}$, for k 2 [K 1], $W^{(k)}$ 2 $\mathrm{R}^{d_h\ \ (d_h+d_z)}$, and $W^{(K)}$ 2 $\mathrm{R}^{d_y\ \ (d_h+d_z)}$. d_h is the dimension of hidden representation x_v^k, for node v at layer k 2 [0; K 1]. For a node v 2 V_i, its initial representation x_v^0 is

$$x_v^0 = W^{(0)}\ [x_v jj Agg(z_v)]^\top\ ^\top; \tag{2}$$

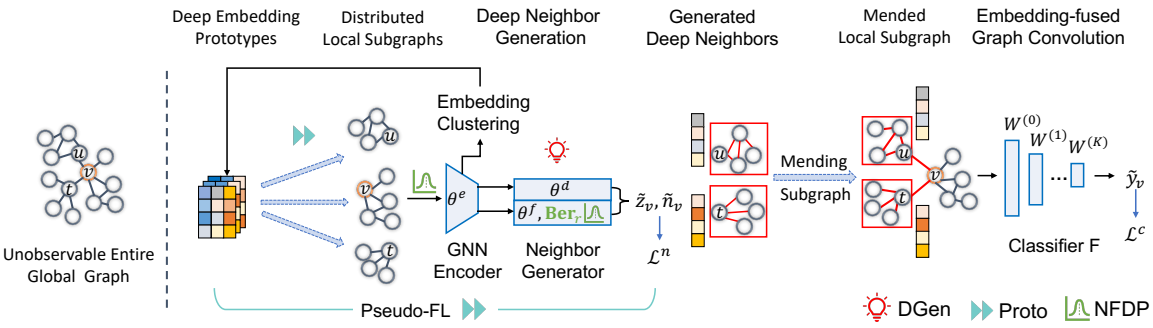

Fig. 3: Overview of the proposed FedDEP framework (with the novel DGen, Proto, and NFDP components highlighted)

and at each layer $k \in [K]$, embedding-fused graph convolution computes v's representation x_v^k as

$$x_v^k = W^{(k)} \left[Agg(\{x_u^{k-1} | u \in G_i^1(v)\}) \,\|\, x_v^{k-1} \right]; \tag{3}$$

where Agg is specified as a mean aggregator, and $\|$ is the concatenating function. After K layers, F outputs the inference label as $\hat{y}_v = \text{Softmax}(x_v^K)$. We formally analyze the correctness of embedding-fused graph convolution as shown below, whose proof is omitted due to the space limit and provided in an online appendix.[1]

Statement 1 (Correctness of embedding-fused graph convolution)**.** *For a node v, at each layer of embedding-fused graph convolution, it aggregates nodes on the impaired ego-graph with the corresponding mended deep neighbor embeddings with separate learnable weights.*

b) ***Inference****:* For a node v on the global graph G without mended deep neighbors, F predicts its label with Eq. (2) and Eq. (3), where all $\tilde{z}_v$'s are simply set to zero vectors.

We further justify that F with K layers of embedding-fused graph convolution in aggregating real neighbors and generated deep neighbors of L-hop local contexts, has a similar capacity as the original one in aggregating an $(K+L)$-hop ego-graph as shown below. The proof is also omitted due to the space limit and provided in an online appendix[1].

Statement 2 (Comparison between embedding-fused graph convolution and original graph convolution)**.** *For a node v, we denote the prediction, computed by K layers of embedding-fused graph convolution on its K-hop impaired ego-graph mended with deep neighbors of L-hop local contexts, as $\hat{y}_v^0$, and the prediction, computed by $(K+L)$ layers of graph convolution on its $(K+L)$-hop ego-graph, as $\hat{y}_v$, where $K; L \in \mathbb{N}$. $\hat{y}_v^0$ and $\hat{y}_v$ encode the same local context of v.*

3) FL for the Joint Model of DGen and F: To achieve a satisfying classification performance for F on nodes drawn from the global graph, it is natural to consider jointly training DGen and F through FL. Within the joint model, DGen can enrich local nodes' neighborhoods to approximate the complete ones on the unobservable global graph, for F to conduct accurate and generalized node classification, while F can supply DGen with task-oriented supervision.

The FL training on the joint model can boil down to two parts, one is learning from a client's local subgraph, *i.e.*, local deep neighbor reconstruction, and the other is learning from others, *i.e.*, cross-subgraph deep neighbor reconstruction. W.l.o.g., we illustrate how D_i federally trains its DGen_i on node $v \in V_i$ as follows.

a) ***Local Deep Neighbor Reconstruction****:* First of all, D_i pre-computes its local node embeddings $Z_i = \{z_v | v \in G_i\}$ via an L-layer GCN (*e.g.*, GraphSage) trained with mini-batch neighbor sampling, where L is a hyper-parameter.

To locally train a DGen, the corresponding owner, *e.g.*, D_i, constructs the ground truth of local missing deep neighbors by impairing its local graph G_i. Specifically, it impairs G_i by hiding its $h\%$ nodes V_i^h and corresponding embeddings Z_i^h, and all related links E_i^h. Then D_i retrieves the impaired subgraph $G_i = \{V_i; E_i; X_i\}$ and corresponding remaining node embeddings Z_i, where $V_i = V_i \cap V_i^h$, $E_i = E_i \cap E_i^h$, $X_i = X_i \cap X_i^h$, and $Z_i = Z_i \cap Z_i^h$. The hidden information enables D_i to locally supervise its DGen. Then, D_i computes the local reconstruction loss $L_{i;i}^n$ as

$$L_{i;i}^n = \frac{1}{|V_i|} \sum_{v \in V_i} \Big[{}^{d}L_1^S(\hat{n}_v, n_v) + {}^{f} \sum_{p \in [\hat{n}_v]} \min_{u \in \mathcal{N}_i(v)} (\|\hat{z}_v^p - z_u\|_2^2) \Big];$$

where $\hat{z}_v^p$ is the p-th generated embeddings in $\hat{z}_v$.

b) ***Cross-subgraph Deep Neighbor Reconstruction****:* Between two clients D_i and D_j, the computation consists of three steps.

1) D_i sends the generated $\hat{z}_v$ to server S for broadcasting.
2) Then, every D_j sends a set of $|\hat{z}_v|$ closest embeddings $z_j \in Z_j$ back to the server, and the server forwards all z_j's to D_i. For $z_j^p \in z_j; p \in [|\hat{z}_v|]$, $z_j^p = \arg\min_{z_u \in Z_j} \|\hat{z}_v^p - z_u\|_2^2$.
3) Finally, on receiving all z_j's, D_i computes the cross-subgraph deep neighbor reconstruction loss between D_i and D_j as $L_{i;j}^n = {}^{n}\frac{1}{|V_i|} \sum_{v \in V_i} \sum_{p \in [|\hat{z}_v|]} (\|\hat{z}_v^p - z_j^p\|_2^2)$, where ^{n} is a constant.

The system minimizes the following aggregated loss function to obtain FL-trained DGen models and the final global classifier F.

$$L^* = \frac{1}{M} \Big(\sum_{i \in [M]} L_i^c + \sum_{j \in [M]} L_{i;j}^n \Big); \tag{4}$$

[1]Appendix accessible in https://anonymous.4open.science/r/FedDEP-6F08/.

where every L_i^c is the cross-entropy loss as

$$L_i^c = l(W; \mathcal{G}_i; Y_i) = \frac{1}{|V_i|} \sum_{v \in V_i} [y_v \log \hat{y}_v + (1 - y_v) \log (1 - \hat{y}_v)]; \quad (5)$$

where $\mathcal{G}_i$ is the subgraph mended by generated deep neighbors, and Y_i is local node label set.

B. Efficiency Elevation through Proto

When the system conducts FL over the joint model of DGen and F, it encounters significant overhead regarding intra-client computations for closest potential neighbor search and inter-client communications for frequent gradient/embedding exchange. To fundamentally reduce both costs, we propose pseudo-FL with deep neighbor prototype generation. We term this combination as Proto.

1) Deep Neighbor Prototype Generation: Technically, every client D_i locally groups pre-computed embeddings Z_i into C clusters by a clustering function. Then, D_i obtains its prototype set as $Z_i^0 = \{mean(z_v | v \in V_i; z_v \text{ in cluster } c) | c \in [C]\}$.

Subsequently, the cross-subgraph prototype reconstruction loss L_i^n is computed on the prototype sets instead of the original node sets as

$$L_i^n = \frac{1}{|V_i|} \sum_{v \in V_i} [\alpha L_1^S(\hat{n}_v - n_v) + \beta \sum_{p \in [\hat{n}_v]} \min_{u \in \mathcal{N}_i; z'_u \in Z'_i} (\|z_v^p - z'_u\|_2^2) + \gamma \sum_{j \in [M] \setminus \{i\}} \sum_{p \in [\hat{n}_v]} \min_{z'_u \in Z'_j} (\|z_v^p - z'_u\|_2^2)]; \quad (6)$$

where α's are constants. Hence, the intra-client search space in computing the cross-subgraph deep neighbor reconstruction loss reduces from $|V|$ nodes to M sets of C prototypes.

The FL training of the joint model with prototyping is to minimize the following objective function

$$L = \frac{1}{M} \sum_{i \in [M]} L_i = \frac{1}{M} \sum_{i \in [M]} (L_i^n + L_i^c); \quad (7)$$

where L_i^c is computed by Eq. (5) with substituting $\mathcal{G}_i$ to the deep neighbor prototype mended subgraph.

2) Pseudo-FL with Cross-subgraph Prototype Generation: To further reduce communication costs without forbidding clients from learning across the system, we propose pseudo-FL based on deep neighbor prototype generation.

In pseudo-FL, each D_i sends Z_i^0 across the system before the FL process. For every D_i, after obtaining $Z^0 = \{Z_j^0 | j \in [M]\}$, it can conduct the FL process for DGen by *locally* computing the cross-subgraph deep neighbor reconstruction loss L_i in Eq. (7) with zero inter-client communication. Then, among deep neighbor prototype mended subgraphs, the system conducts a generic FL (*e.g.*, FedAvg) to attain the final classifier by minimizing L in Eq. (7).

Efficiency Analysis. The main difference between FedSage/FedDEP and generic FL frameworks (*e.g.*, FedAvg) is the additional learning of neighbor generators. Therefore, we analyze the additional overhead caused by the FL training of the neighbor generators for three different frameworks, as shown in Table I. The additional computation complexity for FedDEP is decreased from FedSage due to the reduction in the generated dimension (for real-world datasets, d_x can be a few thousand [46], while for FedDEP, d_z is often less than 300). By prototyping deep neighbors into C clusters, where C is an independently determined cluster number that can be rather small such as 10, the computation complexity of FedDEP is further significantly decreased. Communication-wise, the cost of FedSage is dominated by the size of the generator ($|\theta|$), which can be as large as 3MB even for a simple two-layer GCN model. FedDEP without PROTO reduces the cost by only sharing the deep neighbors. With pseudo-FL, FedDEP can further reduce the communication to zero by sharing $O(MCd_z)$ data ahead, for training DGen.

TABLE I: The additional overhead caused by the FL training of neighbor generators. For simplicity, our analysis is for every round of updating one generator for one node.

FL scheme	comp.	comm./epoch	total comm.
FedSage	$O(\lvert V\rvert \hat{n}_v d_x)$	$O(M\lvert\theta\rvert \lvert\hat{n}_v^K\rvert)$	$O(E_g M\lvert\theta\rvert \lvert\hat{n}_v^K\rvert)$
FedDEP $_{\text{w/o Proto}}$	$O(\lvert V\rvert \hat{n}_v d_z)$	$O(M\hat{n}_v d_z)$	$O(E_g M\hat{n}_v d_z)$
FedDEP	$O(MC^2 d_z)$	0	$O(MCd_z)$

C. Privacy Guarantees through NFDP

We theoretically analyze the edge-LDP property of FedDEP achieved by our novel noise-free DP mechanism (NFDP). Even without explicitly injecting random noises into the original local neighborhoods, our proposed framework sustains strong privacy protections by rigorous edge-LDP.

Theorem 1 (Noise-free edge-LDP of FedDEP)**.** *For a distributed subgraph system, on each subgraph, given every node's L-hop ego-graph with its every L-1 hop neighbors of degrees by at least D, FedDEP unifies all subgraphs in the system to federally train a joint model of a classifier and a cross-subgraph deep neighbor generator. By learning from deep neighbor embeddings that are obtained from locally trained GNNs in N epochs of mini-batch training with a sampling size for each hop as d, FedDEP achieves $(\log(1 + r(e^{\varepsilon} - 1); r\tilde{\ })$-edge-LDP, where*

$$\varepsilon = \min\{LN\varepsilon'; LN\varepsilon' \frac{(e^{\varepsilon'} - 1)}{e^{\varepsilon'} + 1} + \varepsilon' U \sqrt{2LN}\};$$
$$\tilde{\ } = (1 - \ )^{LN}(1 - \ '); \quad \ ' \in [0; 1];$$

and $U = \min\{\sqrt{\ln(e + \frac{\varepsilon' \sqrt{LN}}{\ '})}; \sqrt{\ln(\frac{1}{\ '})}\}$. r is the expected value of the Bernoulli sampler in DGen. When $d < D$, $(\varepsilon'; \ )$ are tighter than $(\ln \frac{D+1}{D+1-d}; \frac{d}{D})$; when $d = D$, $(\varepsilon'; \ )$ are tighter than $(d \ln \frac{D+1}{D}; 1 - (\frac{D-1}{D})^d)$. Both pairs of $(\varepsilon'; \ )$ serve as the lower bounds of the edge-LDP protection under the corresponding cases.

Since both ε and $\tilde{\ }$ are simultaneously affected by the sampling size of local model training, for simplicity, we choose $\tilde{\ }$ to evaluate privacy costs in our experiments. The proof of the Theorem will follow general noise-free DP [26], the rule of the composition of DP mechanisms [47], and privacy amplification by subsampling [48]. Due to the space limit, the detailed proof is omitted and provided in an online appendix[1].

TABLE II: Statistics of the datasets and the synthesized distributed subgraph systems. $|V_i|$ and $|E_i|$ rows show the averaged numbers of nodes and links in all subgraphs, and E shows the total number of missing cross-subgraph links.

Dataset	Cora			Citeseer			PubMed			MSAcademic		
$(\|V\|;\|E\|)$	(2708, 5278)			(3327, 4552)			(19717, 44324)			(18333, 81894)		
$(d_x;\|Y\|)$	(1433, 7)			(3703, 6)			(500, 3)			(6805, 15)		
M	3	5	10	3	5	10	3	5	10	3	5	10
$\|V_i\|$	903	542	271	1109	665	333	6572	3943	1972	6111	3667	1833
$\|E_i\|$	1594	945	437	1458	866	431	13251	7901	3500	24300	13949	5492
E	496	552	912	178	224	247	4570	4818	9323	8995	12149	26973
$E/\|E\|$	9.40%	10.46%	17.28%	3.91%	4.92%	5.43%	10.31%	10.87%	21.03%	10.98%	14.84%	32.94%

Discussions. Proto does not theoretically tighten the privacy bound of edge-LDP. However, unlike individual node features or node embeddings, prototypes in Proto are aggregated data and do not have a one-to-one correspondence with individual nodes. Thus, Proto not only benefits FL efficiency, but also enhances the empirical privacy protection of FedDEP.

V. Experiments

We conduct experiments on four real-world graph datasets to verify the benefits of FedDEP under different scenarios, with in-depth component studies for DGen, Proto, and NFDP.

A. Experimental Settings

We synthesize the distributed subgraph system with four widely used graph datasets, *i.e.*, Cora [49], Citeseer [49], PubMed [50], and MSAcademic [46]. We follow FedSage [14] to synthesize the distributed subgraph system using the Louvain Algorithm. We split every dataset into 3, 5, and 10 subgraphs of similar sizes, and the statistics of these datasets are presented in Table II.

We specify the GNN as a two-layer GraphSage model using the mean aggregator [27]. For each layer of GraphSage, it samples 5 nodes. We use batch size 32 and set training epochs to 50. The same parameters are used for F with embedding-fused graph convolution. The training-validation-testing ratio is 60%-20%-20% and the hyper-parameters for weighting loss values, *i.e.*, s, s, and s, are set to 1. The graph impairing ratio h is set to 0.5 for all cases. Optimization is done using SGD with a learning rate of 0.1. We fix embedding dimension d_z as 128 for Cora, 64 for CiteSeer, 256 for both PubMed and MSAcademic, based on grid search over $\{64,128,256\}$. We implement FedDEP on the FederatedScope platform [51] in Python. The applied clustering function is DEC [52]. We execute all experiments on a server with 8 NVIDIA GeForce GTX 1080 Ti GPUs.[2]

We conduct comprehensive performance evaluations of FedDEP by comparing the following baselines and ablations:

Global: A GraphSage model trained on the entire global graph without missing links (providing the performance upper bound);

Local: A set of GraphSage models trained on individual subgraphs;

FedAvg (FedDEP without DGen/Proto/NFDP): A GraphSage model trained across subgraphs by FedAvg;

FedGNN: A GraphSage model trained across subgraphs by FedGNN [19];

FedGraph: A GraphSage model trained across subgraphs by FedGraph [36];

FedSage: A GraphSage model trained across subgraphs by FedSage+ [14];

FedDEP w/o DGen: FedDEP without deep neighbor generation;

FedDEP w/o Proto: FedDEP without pseudo-FL or embedding prototype;

FedDEP w/o NFDP: FedDEP trained with DPSGD instead of noise-free edge LDP;

FedDEP: The full FedDEP model with DGen, Proto and NFDP.

Cluster numbers in FedDEP are chosen by grid search over $C \in \{3,5,10,15,20\}$ and will be studied in detail in Section V-D. For FedGNN and FedDEP $_{w/o NFDP}$, we fix their as 2.0 to achieve the same level of edge-LDP protection as other variations of FedDEP. We provide results with different privacy budgets in Section V-E.

The metric we use is node classification accuracy on the queries sampled from the testing nodes on the global graph. For the global and FL-trained models, we report the average accuracy over three random repetitions. For the locally trained models, the scores are further averaged across local models. Besides averaged accuracies, we also provide the corresponding standard deviations.

B. Overall Performances

We conduct comprehensive ablation experiments to verify the significant elevations brought by our proposed techniques, as shown in Table III. The most exciting observation is that besides outperforming local models by an average of 27.13%, FedDEP distinctly elevates the performance of FedGNN by at most 2.13%, and FedSage by at most 3.84%, even by requiring zero communication during the FL of the generators. Notably, similar to FedSage, FedDEP exhibits its capacities in elevating beyond the global classifier which is supposed to provide the performance upper bound, possibly due to the additional model robustness brought by the missing neighbor generators. As

[2]Code is provided in https://anonymous.4open.science/r/FedDEP-6F08/.

TABLE III: Node classification results. The top two models are highlighted (except for Global).

Model	Cora						Citesser					
	M=3		M=5		M=10		M=3		M=5		M=10	
Local	0.5776	0.0250	0.4486	0.1079	0.4334	0.0832	0.6541	0.0284	0.5802	0.0560	0.4200	0.1102
FedAvg	0.8571	0.0146	0.8555	0.0143	0.8528	0.0195	0.7646	0.0109	0.7496	0.0102	0.7350	0.0079
FedGNN	0.8823	0.0166	0.8670	0.0100	0.8675	0.0106	0.7850	0.0133	0.7927	0.0141	0.7823	0.0109
FedGraph	0.8693	0.0015	0.8602	0.0042	0.8507	0.0103	0.7720	0.0334	0.7834	0.0205	0.7633	0.0123
FedSage	0.8708	0.0090	0.8790	0.0093	0.8588	0.0099	0.7818	0.0023	0.7805	0.0169	0.7656	0.0102
FedDEP $_{\text{w/o DGen}}$	0.8718	0.0065	0.8717	0.0039	0.8583	0.0069	0.7616	0.0057	0.7806	0.0048	0.7413	0.0051
FedDEP $_{\text{w/o Proto}}$	**0.8911**	**0.0034**	**0.8900**	**0.0028**	**0.8916**	**0.0171**	**0.8107**	**0.0176**	**0.7995**	**0.0094**	**0.8080**	**0.0096**
FedDEP $_{\text{w/o NFDP}}$	0.8883	0.0178	0.8703	0.0148	0.8747	0.0090	0.7846	0.0176	0.7913	0.0106	0.7882	0.0164
FedDEP	**0.8894**	**0.0164**	**0.8883**	**0.0107**	**0.8801**	**0.0087**	**0.7927**	**0.0141**	**0.7940**	**0.0164**	**0.8040**	**0.0219**
Global			0.8955	0.0043					0.7741	0.0045		

Model	PubMed						MSAcademic					
	M=3		M=5		M=10		M=3		M=5		M=10	
Local	0.8287	0.0079	0.7879	0.0322	0.4364	0.1120	0.7906	0.0114	0.7713	0.0992	0.5445	0.0724
FedAvg	0.7149	0.0121	0.7260	0.0023	0.6954	0.0259	0.6986	0.0019	0.6908	0.0201	0.6705	0.0123
FedGNN	0.9009	0.0059	0.8854	0.0047	0.8576	0.0063	0.9403	0.0017	0.9396	0.0004	**0.9362**	**0.0003**
FedGraph	0.8921	0.0022	0.8774	0.0035	0.8581	0.0068	0.9311	0.0016	0.9225	0.0074	0.9244	0.0054
FedSage	0.8877	0.0077	0.8794	0.0031	0.8639	0.0080	0.9359	0.0005	0.9414	0.0006	0.9314	0.0009
FedDEP $_{\text{w/o DGen}}$	0.8440	0.0008	0.8553	0.0075	0.8273	0.0104	0.9434	0.0006	0.9416	0.0009	0.9331	0.0008
FedDEP $_{\text{w/o Proto}}$	**0.9090**	**0.0051**	**0.8885**	**0.0020**	**0.8697**	**0.0039**	**0.9504**	**0.0039**	**0.9455**	**0.0009**	**0.9362**	**0.0006**
FedDEP $_{\text{w/o NFDP}}$	0.9020	0.0074	0.8819	0.0013	0.8605	0.0021	0.9406	0.0007	0.9387	0.0011	0.9339	0.0004
FedDEP	**0.9039**	**0.0069**	**0.8872**	**0.0029**	**0.8662**	**0.0031**	**0.9452**	**0.0012**	**0.9422**	**0.0012**	0.9351	0.0017
Global			0.8996	0.0005					0.9597	0.0006		

shown in the results of CiteSeer in Table III, FedDEP even excels the global model by at most 2.99%.

Experimental results of comparing FedDEP with FedSage and FedDEP $_{\text{w/o DGen}}$ justify the necessity of generating multi-hop cross-subgraph neighbors. Specifically, FedDEP exceeds FedSage by 1.27% on average, and DGen improves FedDEP by 2.49%. Though Proto can cause slight accuracy loss, *i.e.*, 0.52% on average between FedDEP and FedDEP $_{\text{w/o Proto}}$, it both benefits the efficiency (as shown in Figure 6 and reduces the empirical risks of privacy leakage [24].

Regarding the percentages of missing links in Table II, it is obvious that the more missing information in the system, the more likely a larger performance elevation from DGen can be brought to vanilla FedAvg and FedDEP $_{\text{w/o DGen}}$. For the MSAcademic dataset with 10 clients, we infer the reason for FedGNN to slightly exceed FedDEP as the significant amount of missing inter-subgraph links (32.94%). Under this circumstance, when FedDEP further abstracts shared information through Proto, performance degeneration can be the result. However, even in this difficult scenario, the generation of prototyped deep neighbors can still help FedDEP to clearly outperform FedAVG and FedSage.

Under similar privacy protection, FedDEP on average exceeds FedGNN and FedDEP $_{\text{w/o NFDP}}$ by 0.76% and 0.61%, respectively. Without Proto, FedDEP $_{\text{w/o Proto}}$ on average outperforms FedGNN and FedDEP $_{\text{w/o NFDP}}$ by 1.28% and 1.13%, respectively. Such gaps justify the advantageous privacy-utility trade-off of our novel NFDP technique over DPSGD with noise injection.

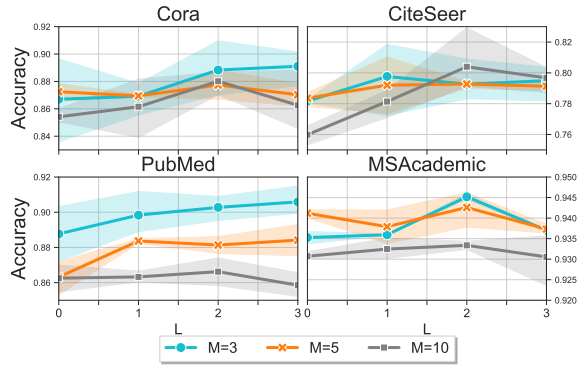

Fig. 4: Component study for DGen in FedDEP with different depths L of generated neighbor embeddings on four datasets with different M's. L=0 is basically FedSage.

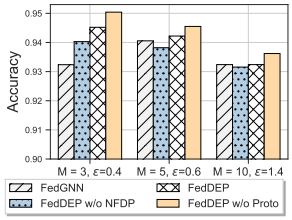

Fig. 5: Component study for NFDP with different levels of edge-LDP privacy protection on MSAcademic with different M's. Corresponding ε's are provided.

C. Component Study of DGen

We conduct in-depth studies for DGen with varying depth L of the generated neighbors in FedDEP. As can be seen in Fig. 4, L controls the amount of neighborhood information exchanged within the system, and positive L can constantly

TABLE IV: Component study for Proto in FedDEP with varying cluster numbers C on four datasets with different M's.

Model	Cora jYj=7						Citesser jYj=6					
	M=3		M=5		M=10		M=3		M=5		M=10	
FedAvg	0.8571	0.0146	0.8555	0.0143	0.8528	0.0195	0.7646	0.0109	0.7496	0.0102	0.7350	0.0079
FedDEP $_{\text{w/o Proto}}$	**0.8911**	**0.0034**	**0.8900**	**0.0028**	**0.8916**	**0.0171**	**0.8107**	**0.0176**	**0.7995**	**0.0094**	**0.8080**	**0.0096**
FedDEP w/ C=3	0.8807	0.0147	0.8670	0.0049	0.8686	0.0043	0.7633	0.0075	0.7873	0.0141	0.7827	0.0173
FedDEP w/ C=5	0.8801	0.0143	0.8569	0.0115	0.8593	0.0109	**0.7927**	**0.0141**	0.7904	0.0281	0.7886	0.0108
FedDEP w/ C=10	0.8851	0.0028	0.8736	0.0215	0.8659	0.0153	0.7769	0.0187	**0.7940**	**0.0154**	**0.8040**	**0.0219**
FedDEP w/ C=15	**0.8894**	**0.0164**	**0.8883**	**0.0107**	**0.8801**	**0.0087**	0.7873	0.0116	0.7913	0.0210	0.7963	0.0150
FedDEP w/ C=20	0.8883	0.0199	0.8703	0.0072	0.8599	0.0090	0.7850	0.0077	0.7909	0.0215	0.7724	0.0182
Model	PubMed jYj=3						MSAcademic jYj=15					
	M=3		M=5		M=10		M=3		M=5		M=10	
FedAvg	0.7149	0.0121	0.7260	0.0023	0.6954	0.0259	0.6986	0.0019	0.6908	0.0201	0.6705	0.0123
FedDEP $_{\text{w/o Proto}}$	**0.9090**	**0.0051**	**0.8885**	**0.0020**	**0.8697**	**0.0039**	**0.9504**	**0.0039**	**0.9455**	**0.0009**	**0.9362**	**0.0006**
FedDEP w/ C=3	**0.8996**	**0.0069**	0.8862	0.0097	0.8650	0.0183	0.9354	0.0005	0.9365	0.0010	0.9314	0.0010
FedDEP w/ C=5	0.8929	0.0086	**0.8872**	**0.0029**	0.8652	0.0024	0.9353	0.0008	**0.9422**	**0.0012**	**0.9351**	**0.0017**
FedDEP w/ C=10	0.8995	0.0047	0.8817	0.0054	0.8642	0.0079	0.9353	0.0007	0.9352	0.0006	0.9313	0.0008
FedDEP w/ C=15	0.8961	0.0113	0.8804	0.0038	**0.8662**	**0.0031**	**0.9452**	**0.0012**	0.9393	0.0013	0.9302	0.0008
FedDEP w/ C=20	0.8917	0.0043	0.8781	0.0061	0.8580	0.0072	0.9351	0.0007	0.9353	0.0008	0.9313	0.0004

elevate testing accuracy, compared with only exchanging neighbor features in FedSage (L=0). Across different datasets, the optimal L is usually around 2. When the dataset has too many missing links between subgraphs, such as when M=10, large L can introduce more biased deep neighbor embeddings, leading to worse performances.

D. Component Study of Proto

We compare the downstream task performance of FedDEP under different numbers of cluster C in Proto. Table IV shows that choosing a proper C, which controls how abstract the exchanged information is within the system, can constantly elevate the final testing accuracy. Across different datasets, when C is chosen around the number of classes, the performance is usually good. C being too small like 3 or too large like 20 can result in slight performance drops, but FedDEP is in general insensitive to C in a wide range.

E. Component Study of NFDP

We compare models' utility under the same differential privacy guarantees for noise-free frameworks (FedDEP and FedDEP $_{\text{w/o Proto}}$) and noise-injected frameworks (FedGNN and FedDEP $_{\text{w/o NFDP}}$), as shown in Fig. 5. Specifically, in MSAcademic dataset, when M=3, we have d=5, D=15, ε=0.4, and δ=4.2; when M=5, we have d=5, D=10, ε=0.6, and δ=2.1; when M=10, we have d=5, D=3, ε=1.4, and δ=0.4. As shown in Fig. 5, our NFDP technique always outperforms the noise-injected counterparts when similar levels of edge-LDP are achieved, which empirically justifies the superior utility-privacy trade-off of NFDP compared to gradients perturbation-based approaches such as DPSGD.

F. Convergence Analysis

For the Cora dataset with five data owners, we visualize testing accuracy, loss convergence, and runtime along 100 epochs in obtaining F with Global, Local, FedAvg, FedSage, FedDEP $_{\text{w/o Proto}}$, and FedDEP. The results are presented in Fig. 6. Both FedDEP and FedDEP $_{\text{w/o Proto}}$ consistently achieve satisfactory convergence with rapidly improved testing accuracy. Regarding runtime, even though the classifiers from FedDEP $_{\text{w/o Proto}}$ and FedDEP learn from distributed mended subgraphs, they do not consume observable more training time compared to FedAvg. Compared to FedSage, FedDEP and its variation reduce the dimension of mended information, and thus save non-neglectable training time. Thanks to Proto, FedDEP consumes far less time than FedDEP $_{\text{w/o Proto}}$ and only slightly more time compared to Global and FedAvg.

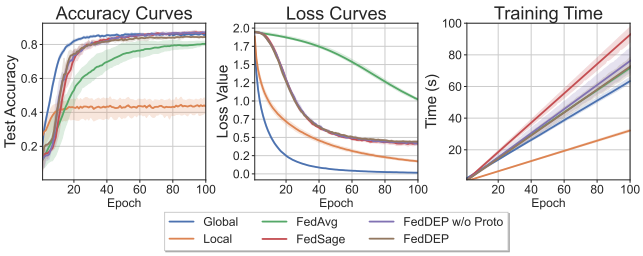

Fig. 6: Training curves of different frameworks on Cora dataset with M=5. (Best viewed in color.)

VI. Conclusion

In this work, we study subgraph FL by comprehensively tackling the unique challenges of utility, efficiency, and privacy. Promising future directions of FedDEP include the employment of more powerful GNN models such as based on graph transformers, pruning of unnecessary graph generation such as based on model uncertainty, stronger privacy protection such as regarding node-LDP based on the integration of noise-free DP and DPSGD, and application studies on other real-world graph datasets.

ACKNOWLEDGEMENT

This research was partially supported by the National Science Foundation under Award Number 2319449 and Award Number 2312502, as well as the National Institute Of Diabetes And Digestive And Kidney Diseases of the National Institutes of Health under Award Number K25DK135913. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily represent the views of the National Science Foundation, National Institutes of Health, or the U.S. government.

REFERENCES

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[2] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *SIGIR*, 2020.
[3] X. Wang, Y. Ma, Y. Wang, W. Jin, X. Wang, J. Tang, C. Jia, and J. Yu, "Traffic flow prediction via spatial temporal graph neural network," in *WWW*, 2020.
[4] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *ICLR*, 2019.
[5] E. Choi, M. T. Bahadori, L. Song, W. F. Stewart, and J. Sun, "Gram: graph-based attention model for healthcare representation learning," in *WWW*, 2017.
[6] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *WWW*, 2018.
[7] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," in *ICLR*, 2020.
[8] D. Zheng, M. Wang, Q. Gan, Z. Zhang, and G. Karypis, "Learning graph neural networks with deep graph library," in *WWW*, 2020.
[9] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, vol. 10, 2017.
[10] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *TIST*, vol. 10, no. 2, pp. 1–19, 2019.
[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.
[12] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, L. Feng, T. Chen, H. Yu, and Q. Yang, "Fedvision: An online visual object detection platform powered by federated learning," in *AAAI*, 2020.
[13] B. Y. Lin, C. He, Z. Zeng, H. Wang, Y. Huang, M. Soltanolkotabi, X. Ren, and S. Avestimehr, "Fednlp: A research platform for federated learning in natural language processing," in *Findings of the Association for Computational Linguistics: NAACL*, 2021.
[14] K. Zhang, C. Yang, X. Li, L. Sun, and S. M. Yiu, "Subgraph federated learning with missing neighbor generation," in *NeurIPS*, 2021.
[15] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *S&P*, 2019.
[16] X. Luo, Y. Wu, X. Xiao, and B. C. Ooi, "Feature inference attack on model predictions in vertical federated learning," in *ICDE*, 2021.
[17] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *NeurIPS*, 2019.
[18] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" in *NeurIPS*, 2020.
[19] C. Wu, F. Wu, L. Lyu, T. Qi, Y. Huang, and X. Xie, "A federated graph neural network framework for privacy-preserving personalization," *Nature Communications*, 2022.
[20] D. Chai, L. Wang, K. Chen, and Q. Yang, "Secure federated matrix factorization," *IEEE Intelligent Systems*, vol. 36, no. 5, pp. 11–20, 2020.
[21] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *CCS*, 2016.
[22] Q. Zhu, C. Yang, Y. Xu, H. Wang, C. Zhang, and J. Han, "Transfer learning of graph neural networks with ego-graph information maximization," in *NeurIPS*, 2021.
[23] M. Tang, C. Yang, and P. Li, "Graph auto-encoder via neighborhood wasserstein reconstruction," in *ICLR*, 2022.
[24] Y. Tan, G. Long, L. Liu, T. Zhou, Q. Lu, J. Jiang, and C. Zhang, "Fedproto: Federated prototype learning across heterogeneous clients," in *AAAI*, 2022.
[25] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating synthetic decentralized social graphs with local differential privacy," in *SIGSAC*, 2017.
[26] L. Sun and L. Lyu, "Federated model distillation with noise-free differential privacy," in *IJCAI*, 2021.
[27] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017.
[28] H. Peng, H. Li, Y. Song, V. Zheng, and J. Li, "Differentially private federated knowledge graphs embedding," in *CIKM*, 2021.
[29] M. Chen, W. Zhang, Z. Yuan, Y. Jia, and H. Chen, "Fede: Embedding knowledge graphs in federated setting," in *IJCKG*, 2020.
[30] K. Zhang, Y. Wang, H. Wang, L. Huang, C. Yang, and L. Sun, "Efficient federated learning on knowledge graphs via privacy-preserving relation embedding aggregation," in *Findings of EMNLP*, 2022.
[31] Z. Gu, K. Zhang, G. Bai, L. Chen, L. Zhao, and C. Yang, "Dynamic activation of clients and parameters for federated learning over heterogeneous graphs," in *ICDE*, 2023.
[32] H. Xie, J. Ma, L. Xiong, and C. Yang, "Federated graph classification over non-iid graphs," in *NeurIPS*, 2021.
[33] S. Liu, R. Ying, H. Dong, L. Li, T. Xu, Y. Rong, P. Zhao, J. Huang, and D. Wu, "Local augmentation for graph neural networks," in *ICML*, 2022.
[34] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *ICLR*, 2020.
[35] K. Zhang, H. Xie, Z. Gu, X. Li, L. Sun, S. M. Yiu, Y. Yao, and C. Yang, "Subgraph federated learning over heterogeneous graphs," in *FedGraph-CIKM*, 2022.
[36] F. Chen, P. Li, T. Miyazaki, and C. Wu, "Fedgraph: Federated graph learning with intelligent sampling," *IEEE TPDS*, 2021.
[37] C. Dwork, "Differential privacy," in *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33*. Springer, 2006, pp. 1–12.
[38] C. Yang, H. Wang, K. Zhang, L. Chen, and L. Sun, "Secure deep graph generation with link differential privacy," in *IJCAI*, 2021.
[39] J. Zhou, C. Chen, L. Zheng, X. Zheng, B. Wu, Z. Liu, and L. Wang, "Vertically federated graph neural network for privacy-preserving node classification," in *IJCAI*, 2021.
[40] W. Lu and G. Miklau, "Exponential random graph estimation under differential privacy," in *KDD*, 2014, pp. 921–930.
[41] F. Ahmed, A. X. Liu, and R. Jin, "Publishing social network graph eigenspectrum with privacy guarantees," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 892–906, 2019.
[42] Q. Xiao, R. Chen, and K.-L. Tan, "Differentially private network data release via structural inference," in *KDD*, 2014.
[43] J. Imola, T. Murakami, and K. Chaudhuri, "Differentially private triangle and 4-cycle counting in the shuffle model," in *SIGSAC*, 2022.
[44] W. Lin, B. Li, and C. Wang, "Towards private learning on decentralized graphs with local differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2936–2946, 2022.
[45] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Workshop of NeurIPS*, 2016.
[46] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," *arXiv preprint arXiv:1811.05868*, 2018.
[47] P. Kairouz, S. Oh, and P. Viswanath, "The composition theorem for differential privacy," in *ICML*, 2015.
[48] B. Balle, G. Barthe, and M. Gaboardi, "Privacy profiles and amplification by subsampling," *Journal of Privacy and Confidentiality*, vol. 10, no. 1, 2020.
[49] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.
[50] G. Namata, B. London, L. Getoor, and B. Huang, "Query-driven active surveying for collective classification," in *MLG workshop*, 2012.
[51] Z. Wang, W. Kuang, Y. Xie, L. Yao, Y. Li, B. Ding, and J. Zhou, "Federatedscope-gnn: Towards a unified, comprehensive and efficient package for federated graph learning," in *KDD*, 2022.
[52] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *ICML*, 2016.