Deep Efficient Private Neighbor Generation for Subgraph Federated Learning

Ke Zhang* Lichao Sun† Bolin Ding‡ Siu Ming Yiu§ Carl Yang¶

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 justify the clear benefits of proposed techniques.

Keywords: Federated Learning, Graph Mining, Neighbor Generation, Efficiency, Privacy Protection

1 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 [13], e-commerce [10], transportation [27], bioinformatics [30], and healthcare [6]. The power of GNNs benefits from training on real-world graphs with millions to billions of nodes and links [33]. 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 [26]. Please refer to detailed related work in Appendix C.

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 [31]. Despite its successful applications in domains like computer vision [15] and natural language processing [14] 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 [2, 32]. 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 shown 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 generic FL mechanisms such as FedAvg for GNN training. Targeting this limitation, Zhang et al. propose FedSage [34], 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))[1]. FedSage's success justifies the necessity of completing information in local neighborhoods. Yet it has 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

*cszhangk@connect.hku.hk, ClusterTech Limited.
†lis221@lehigh.edu, Lehigh University.
‡bolin.ding@alibaba-inc.com, Alibaba Group.
§smyiu@cs.hku.hk, The University of Hong Kong.
¶j.carlyang@emory.edu, Emory University.

[1] For simplicity, we refer to FedSage+ in [34] as FedSage.

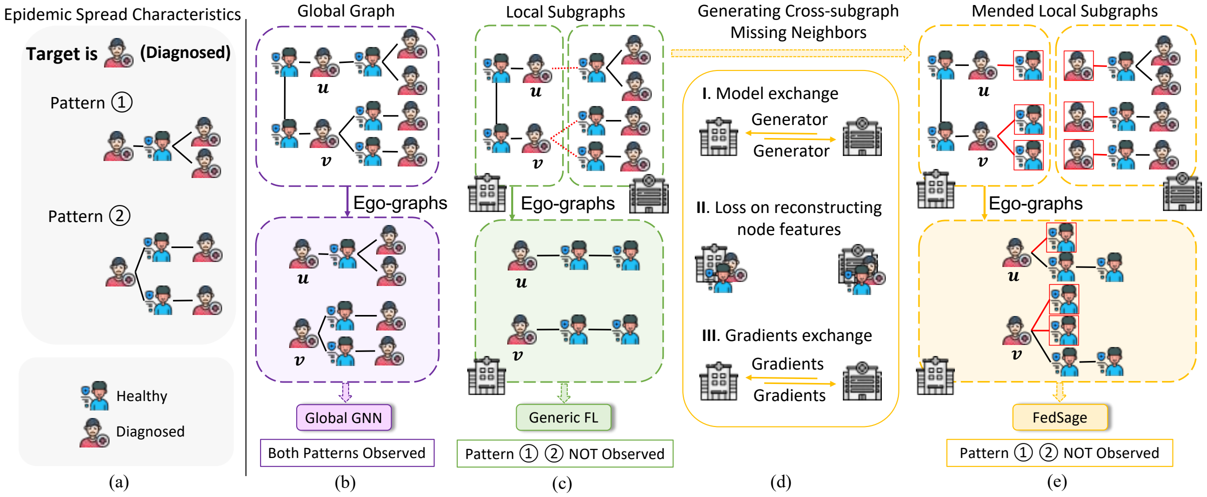

Figure 1: A toy example of modeling the spread of infectious disease in a distributed subgraph FL system. The black lines are the close contact relations between people, and the dashed red lines are the cross-subgraph missing links. Red solid lines are the generated links, and the people figures with 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 healthy neighbors directly contact with many diagnosed ones, or Pattern ②: many healthy neighbors directly 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 recover 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 still fail because the correct patterns require access to deeper missing neighbors.

neighboring nodes. As illustrated in Fig. 1 (e), for predictions where neighbors further than 1 hop away are important, the model will fail. Such limitation is verified by the limited performance gain of FedSage+ over vanilla FedSage without neighbor generator in [34] as well as our empirical results in Table 2.

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. Other existing studies, such as FedGraph [5] and Lumos [19], request the central server to manage the FL process with auxiliary data, which induce unneglectable communication and computation overhead.

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 directly computed from raw data are prone to privacy leakage, such as via inference attacks [16, 18] and reconstruction attacks [8, 36]. Unlike FedSage, FedGNN [29] and FedGSL [35] protect privacy for FL over graphs via noise injection [1, 7]. FedGSL lacks discussions on the privacy protection of graph properties, while FedGNN requests an additional authority to guarantee privacy, 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 subgraph FL setting.

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 GNN embeddings of generated neighbors. The generated embeddings contain information from the target node's multiple hops of neighbors [25,37] and include richer context beyond single node features generated in FedSage. To incorporate the generated deep neighbors into the global GNN classifier training, we propose a novel embedding-fused graph convolution process.

Efficiency-wise: Deep Embedding Prototyping and Pseudo-FL (Proto). To reduce the intra-client computation, we cluster the local 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 [20]. Particularly, we achieve noise-free edge-LDP by transferring noise-free differential privacy from general domains [23] to edge-LDP, without embracing complicated cryptology techniques, explicitly perturbing shared data, or introducing additional roles into the system as previous work [29]. Technically, we incorporate two stages of random sampling into FedDEP, *i.e.*, (1) mini-batching: random neighborhood sampling in each graph convolution layer [9]; 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.

2 Problem Formulation

Federated Learning with Distributed Subgraphs. 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 $V = \bigcup_{i=1}^{M} V_i$. For simplicity, we assume no overlapping nodes shared across data owners, *i.e.*, $V_i \cap V_j = \emptyset$ for any $i \neq j \in [M]$. 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. The system exploits an FL framework to collaboratively learn a global node classifier F on isolated subgraphs in clients, without raw graph data sharing.

3 FedDEP

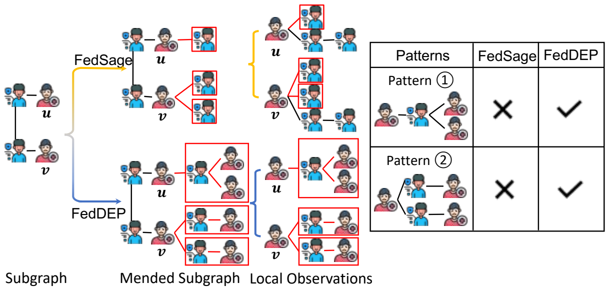

Figure 2: Technical motivation of FedDEP against FedSage. FedDEP generates information of multiple hops of neighbors to provide the subgraph with richer information for local nodes, compared to the direct missing neighbors generated by FedSage. For more details of FedSage, please refer to the background discussion in Appendix D and the FedSage paper [34].

3.1 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 incurs exponential increments for intra-client computation and inter-client communication. To solve this, we propose to leverage the GNN encoder and generate deep embeddings of missing neighbors that captures their multi-hop local contexts in the corresponding subgraphs. Fig. 2 illustrates this technical motivation, regarding the toy example in Fig. 1.

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

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

$$\tilde{n}_v = \sigma(\theta^{d\top} \cdot Enc(G_i^K(v); \theta^e)),$$

$$\tilde{z}_v = Ber_r\left(\sigma\left(\theta^{f\top} \cdot [Enc(G_i^K(v); \theta^e) + \mathbf{N}(0,1)]\right), \tilde{n}_v\right),$$

where Enc is the GNN encoder of DGen, $\tilde{n}_v \in \mathbb{N}$, $\tilde{z}_v \in \mathbb{R}^{\tilde{n}_v \times d_z}$, 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.

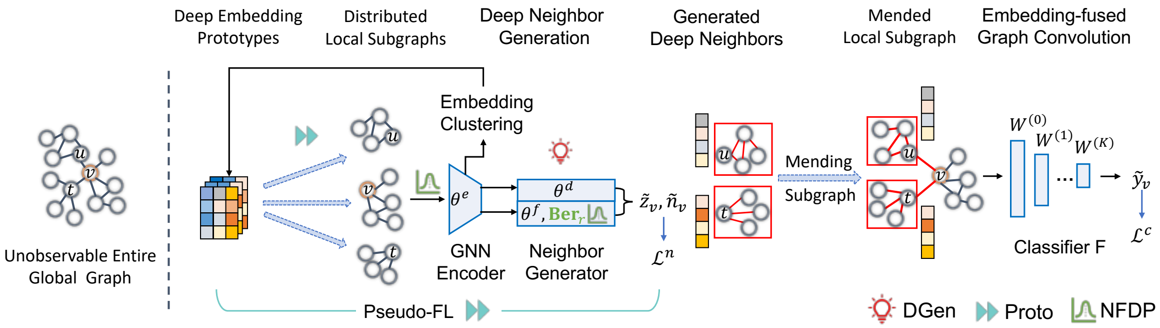

Figure 3: Overview of the proposed FedDEP (with the novel DGen, Proto, and NFDP components highlighted).

In this process, based on the original neighborhood of v, DGen first predicts $\tilde{n}_v$, the number of its missing neighbors n_v, and then samples $\tilde{n}_v$ embedding vectors for them. Sampling allows the whole process to be generative and trained through variational inference for enhanced robustness [12].

Embedding-fused Graph Convolution. On mending local subgraphs with generated deep neighbors, *i.e.*, attaching nodes with deep embeddings, every client obtains its mended local subgraph $\tilde{G}_i = \{V_i, \tilde{E}_i, X_i, \tilde{Z}_i\}$. Yet existing GNNs with vanilla graph convolution process is incapable to incorporate node features with deep embeddings, due to the difference between their feature spaces. To conduct node classification on $\tilde{G}_i$ to fulfill the goal of obtaining a node classifier F, we propose the embedding-fused graph convolution mechanism.

Downstream Classifier F. F is a model of K-layer embedding-fused graph convolution.

For $v \in V$, F integrates v's K-hop mended ego-graph $\tilde{G}_i^K(v)$ on $\tilde{G}_i$. The convolution is achieved by learnable weights $W{=}\{W^{(k)}|k \in [0, K]\}$, where $W^{(0)} \in \mathbb{R}^{d_h \times (d_x+d_z)}$, for $k \in [K\text{-}1]$, $W^{(k)} \in \mathbb{R}^{d_h \times (d_h+d_z)}$, and $W^{(K)} \in \mathbb{R}^{d_y \times (d_h+d_z)}$. d_h is the dimension of hidden representation x_v^k, for node v at layer $k \in [0, K\text{-}1]$.

For a node $v \in V_i$, its initial representation x_v^0 is

$$x_v^0 = \sigma\left(W^{(0)} \times [x_v||Agg(\tilde{z}_v)]^\top\right)^\top,$$

where Agg is specified as a mean aggregator, and $||$ is the concatenating function.

For layer $k \in [K]$, the model computes v's representation x_v^k as

$$x_v^k = \sigma\left(W^{(k)} \times [Agg(\{x_u^{k-1}|u \in G_i^1(v)\})^\top||Agg(\tilde{z}_v)]^\top\right)^\top.$$

After K layers, F outputs the inference label as $\tilde{y}_v = \text{Softmax}(x_v^K)$. We formally analyze the correctness of embedding-fused graph convolution and whose proof is in Appendix A.1.

Statement 3.1. *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.*

Inference. For a node v on the global graph G without mended deep neighbors, F predicts its label by setting all $\tilde{z}_v$'s to zero vectors. We 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 GCN in aggregating an $(K{+}L)$-hop ego-graph as shown below. The proof is provided in Appendix A.2.

Statement 3.2. *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, as $\tilde{y}_v'$, and the prediction, computed by $(K{+}L)$ layers of graph convolution on its $(K{+}L)$-hop ego-graph, as $\tilde{y}_v$, where $K, L \in \mathbb{N}^*$. $\tilde{y}_v'$ and $\tilde{y}_v$ encode the same local context of v.*

FL for the Joint Model of DGen and F. We follow FedSage to jointly train DGen and F via FL with

$$\mathcal{L}_{i,i}^n = \frac{1}{|\bar{V}_i|}\sum_{v \in \bar{V}_i}[\lambda^d L_1^S(\tilde{n}_v - n_v) + \lambda^f \sum_{p \in [\tilde{n}_v]} \min_{u \in \bar{\mathcal{N}}_i(v)}(||\tilde{z}_v^p - z_u||_2^2)],$$

where $\tilde{z}_v^p$ is the p-th generated embeddings in $\tilde{z}_v$, n_v and z_u are the local ground truths retrieved from the hidden information. 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.

3.2 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 it as Proto.

Deep Neighbor Prototype Generation. Technically, every client D_i first locally trains a GNN of the same construction as the final F. Next, D_i retrieves pre-computed local node embeddings Z_i from the local GNN, and groups them into C clusters by a clustering function. Then D_i obtains its prototype set as $Z_i' = \{mean(z_v | v \in V_i, z_v \text{ in cluster } c) | c \in [C]\}$.

Subsequently, the cross-subgraph prototype reconstruction loss $\mathcal{L}_i^n$ is computed on the prototype sets as

$$\mathcal{L}_i^n = \frac{1}{|\bar{V}_i|} \sum_{v \in \bar{V}_i} [\beta^d L_1^S(\tilde{n}_v - n_v) + \beta^f \sum_{p \in [\tilde{n}_v]} \min_{u \in \bar{\mathcal{N}}_i, z_u' \in Z_i'} (||\tilde{z}_v^p - z_u'||_2^2) \\ + \beta^n \sum_{j \in [M] \setminus \{i\}} \sum_{p \in [\tilde{n}_v]} \min_{z_u' \in Z_j'} (||\tilde{z}_v^p - z_u'||_2^2)],$$

where β's are constants.

In this way, the intra-client search space in computing the reconstruction loss reduces from $|V|$ to $M * C$ prototypes.

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

$$\mathcal{L} = \frac{1}{M} \sum_{i \in [M]} \mathcal{L}_i = \frac{1}{M} \sum_{i \in [M]} (\mathcal{L}_i^n + \mathcal{L}_i^c), \tag{3.1}$$

where $\mathcal{L}_i^c$ is the cross-entropy loss computed on deep neighbor prototype mended subgraph.

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 Proto. In pseudo-FL, each D_i sends Z_i' across the system before the FL process. For every D_i, after obtaining $Z' = \{Z_j' | j \in [M]\}$, it can conduct the FL process for DGen by *locally* computing the cross-subgraph deep neighbor reconstruction loss $\mathcal{L}_i$ in Eq. (3.1) with zero inter-client communication. Then, among deep neighbor prototype mended subgraphs, the system conducts FL (*e.g.*, FedAvg) to attain the final classifier by minimizing $\mathcal{L}$ in Eq. (3.1).

Efficiency Analysis. The main difference between FedSage, FedDEP and generic FL frameworks (*e.g.*, FedAvg) is the additional learning of neighbor generators. We analyze the additional overhead caused by the FL training of the neighbor generators for three different frameworks in Table 1.

The computation complexity for FedDEP is decreased from FedSage by the reduction in the generated dimension (for real-world datasets, d_x can be a few thousand [22], while d_z in FedDEP is often less than 300). By prototyping deep neighbors into C clusters, where C 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 generator's size $|\theta|$, which can be as large as 3MB even for a simple two-layer GCN model. FedDEP without Proto (FedDEP $_{/\text{Proto}}$) reduces the cost by sharing deep neighbors. With pseudo-FL, FedDEP cuts the communication to zero by sharing $O(MCd_z)$ data ahead of training DGen.

Table 1: The additional overhead caused by the FL training of neighbor generator (one round of updating a generator for one node).

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

3.3 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 3.1. (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* $(\log(1 + r(e^{\tilde{\varepsilon}}\text{-}1), r\tilde{\delta})$*-edge-LDP, where*

$$\tilde{\varepsilon} = \min\{LN\varepsilon, LN\varepsilon\frac{(e^{\varepsilon} - 1)}{e^{\varepsilon} + 1} + \varepsilon U\sqrt{2LN}\},$$

$$\tilde{\delta} = (1 - \delta)^{LN}(1 - \delta'), \quad \delta' \in [0, 1],$$

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

Since both ϵ and δ are simultaneously affected by the sampling size of local model training, for simplicity, we choose ϵ to evaluate privacy costs in our experiments. The proof of the Theorem follows general noise-free DP [23], the rule of the composition of DP mechanisms [11], and privacy amplification by subsampling [3]. Due to the space limit, the detailed proof is in Appendix B.

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.

4 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.

4.1 Experimental Settings We synthesize the distributed subgraph system with four widely used graph datasets, *i.e.*, Cora [21], CiteSeer [21], PubMed [17], and MSAcademic [22]. We follow FedSage [34] to synthesize the distributed subgraph system using the Louvain Algorithm [4]. We split every dataset into 3, 5, and 10 subgraphs of similar sizes, and due to the space limit, whose statistics are presented in Appendix E.

We specify the GNN as a two-layer GraphSage with mean aggregator [9] and neighbor size 5. Batch size and training epochs are set to 32 and 50. Same parameters are used for F with embedding-fused graph convolution. The train-val-test ratio is 60%-20%-20% and all loss weights are set to 1. The graph impairing ratio h is set to 0.5. SGD optimization is applied with 0.1 learning rate. d_z is 128 for Cora, 64 for CiteSeer, 256 for both PubMed and MSAcademic, based on the grid search over {64,128,256}. We implement FedDEP on the FederatedScope platform [28] in Python. All experiments are on a server with 8 NVIDIA GeForce GTX 1080 Ti GPUs.[2]

We conduct comprehensive performance evaluations of FedDEP by comparing following baselines and ablations: (1) Global: A GraphSage model trained on the entire global graph without missing links (providing the performance upper bound); (2) Local: A set of GraphSage models trained on individual subgraphs; (3) FedAvg (FedDEP without DGen/Proto/NFDP): A GraphSage model trained across subgraphs by FedAvg; (4) FedGNN: A GraphSage model trained across subgraphs by FedGNN [29]; (5) FedGraph: A GraphSage model trained across subgraphs by FedGraph [5]; (6) FedGSL: A GraphSage model trained across subgraphs by FedGSL [35]; (7) FedSage: A GraphSage model trained across subgraphs by FedSage+ [34]; (8) FedDEP $_{\text{w/o DGen}}$: FedDEP without deep neighbor generation; (9) FedDEP $_{\text{w/o Proto}}$: FedDEP without pseudo-FL or embedding prototype; (10) FedDEP $_{\text{w/o NFDP}}$: FedDEP trained with DPSGD instead of noise-free edge LDP; and (11) 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 Section 4.4. For FedGNN and FedDEP $_{\text{w/o NFDP}}$, we fix their σ 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 4.5.

The metric we use is node classification accuracy on the queries sampled from test nodes on the global graph. The reported average accuracy is over three random repetitions. For locally trained models, the scores are further averaged across local models. The corresponding standard deviations are also provided.

4.2 Overall Performances We conduct comprehensive ablation experiments to verify the significant elevations brought by our proposed techniques, as shown in Table 2. 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 shown in the results of CiteSeer in Table 2, 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 Fig. 6) and reduces the empirical risks of privacy leakage [24].

It is obvious that the more missing links, i.e., 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 of FedGNN slightly exceeding FedDEP to be the significant amount of missing inter-subgraph links (32.94%). In this circumstance, when FedDEP further abstracts shared information through Proto, performance degeneration can be the result. 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 over DPSGD with noise injection.

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

Table 2: Node classification results. The top two models are highlighted (except for Global).

Training	M=3	M=5	M=10	M=3	M=5	M=10
Frameworks	**Cora** *Global*: 0.8955±.004			**CiteSeer** *Global*: 0.7741±.005		
Local	0.5776±.025	0.4486±.108	0.4334±.083	0.6541±.028	0.5802±.056	0.4200±.110
FedAvg	0.8571±.015	0.8555±.014	0.8528±.020	0.7646±.011	0.7496±.010	0.7350±.008
FedGNN	0.8823±.017	0.8670±.010	0.8675±.011	0.7850±.013	0.7927±.014	0.7823±.011
FedGraph	0.8693±.002	0.8602±.004	0.8507±.010	0.7720±.033	0.7834±.021	0.7633±.012
FedGSL	0.8633±.001	0.8620±.006	0.8613 ±.040	0.7810±.043	0.7900±.014	0.7567±.011
FedSage	0.8708±.009	0.8790±.009	0.8588±.010	0.7818±.002	0.7805±.017	0.7656±.010
FedDEP $_{\text{w/o DGen}}$	0.8718±.007	0.8717±.004	0.8583±.007	0.7616±.006	0.7806±.005	0.7413±.005
FedDEP $_{\text{w/o Proto}}$	**0.8911±.003**	**0.8900±.003**	**0.8916±.017**	**0.8107±.018**	**0.7995±.009**	**0.8080±.010**
FedDEP $_{\text{w/o NFDP}}$	0.8883±.018	0.8703±.015	0.8747±.009	0.7846±.018	0.7913±.011	0.7882±.016
FedDEP	**0.8894±.016**	**0.8883±.011**	**0.8801±.009**	**0.7927±.014**	**0.7940±.016**	**0.8040±.022**
	PubMed *Global*: 0.8996±.001			**MSAcademic** *Global*: 0.9597±.001		
Local	0.8287±.008	0.7879±.032	0.4364±.112	0.7906±.011	0.7713±.099	0.5445±.072
FedAvg	0.7149±.012	0.7260±.002	0.6954±.026	0.6986±0.002	0.6908±.020	0.6705±.012
FedGNN	0.9009±.006	0.8854±.005	0.8576±.006	0.9403±.002	0.9396±.001	**0.9362±.001**
FedGraph	0.8921±.002	0.8774±.004	0.8581±.007	0.9311±.002	0.9225±.007	0.9244±.005
FedGSL	0.8991±.003	0.8815±.002	0.8592 ±.004	0.9385±.003	0.93060±.004	0.9268±.001
FedSage	0.8877±.008	0.8794±.003	0.8639±.008	0.9359±.001	0.9414±.001	0.9314±.001
FedDEP $_{\text{w/o DGen}}$	0.8440±.001	0.8553±.008	0.8273±.010	0.9434±.001	0.9416±.001	0.9331±.001
FedDEP $_{\text{w/o Proto}}$	**0.9090±.005**	**0.8885±.002**	**0.8697±.004**	**0.9504±.004**	**0.9455±.001**	**0.9362±.001**
FedDEP $_{\text{w/o NFDP}}$	0.9020±.007	0.8819±.001	0.8605±.002	0.9406±.001	0.9387±.001	0.9339±.001
FedDEP	**0.9039±.007**	**0.8872±.003**	**0.8662±.003**	**0.9452±.001**	**0.9422±.001**	0.9351±.002

Table 3: Component study for Proto with varying cluster numbers C on four datasets with different M's.

Training	M=3	M=5	M=10	M=3	M=5	M=10
Frameworks	**Cora** $\|Y\|$=7			**CiteSeer** $\|Y\|$=6		
FedAvg	0.8571±.015	0.8555±.014	0.8528±.020	0.7646±.011	0.7496±.010	0.7350±.008
FedDEP $_{\text{w/o Proto}}$	**0.8911±.003**	**0.8900±.003**	**0.8916±.017**	**0.8107±.018**	**0.7995±.009**	**0.8080±.010**
FedDEP w/ C=3	0.8807±.015	0.8670±.005	0.8686±.004	0.7633±.008	0.7873±.014	0.7827±.017
FedDEP w/ C=5	0.8801±.014	0.8569±.012	0.8593±.011	**0.7927±.014**	0.7904±.028	0.7886±.011
FedDEP w/ C=10	0.8851±.003	0.8736±.022	0.8659±.015	0.7769±.019	**0.7940±.015**	**0.8040±.022**
FedDEP w/ C=15	**0.8894±.016**	**0.8883±.011**	**0.8801±.009**	0.7873±.012	0.7913±.021	0.7963±.015
FedDEP w/ C=20	0.8883±.020	0.8703±.007	0.8599±.009	0.7850±.008	0.7909±.022	0.7724±.018
	PubMed $\|Y\|$=3			**MSAcademic** $\|Y\|$=15		
FedAvg	0.7149±.012	0.7260±.002	0.6954±.026	0.6986±.002	0.6908±.020	0.6705±.012
FedDEP $_{\text{w/o Proto}}$	**0.9090±.005**	**0.8885±.002**	**0.8697±.004**	**0.9504±.004**	**0.9455±.001**	**0.9362±.001**
FedDEP w/ C=3	**0.8996±.007**	0.8862±.010	0.8650±.018	0.9354±.001	0.9365±.001	0.9314±.001
FedDEP w/ C=5	0.8929±.009	**0.8872±.003**	0.8652±.002	0.9353±.001	**0.9422±.001**	**0.9351±.002**
FedDEP w/ C=10	0.8995±.005	0.8817±.005	0.8642±.008	0.9353±.001	0.9352±.001	0.9313±.001
FedDEP w/ C=15	0.8961±.011	0.8804±.004	**0.8662±.003**	**0.9452±.001**	0.9393±.001	0.9302±.001
FedDEP w/ C=20	0.8917±.004	0.8781±.006	0.8580±.007	0.9351±.001	0.9353±.001	0.9313±.001

4.3 Component Study of DGen We conduct in-depth studies for DGen with varying depth L of the generated neighbors in FedDEP. As shown in Fig. 4, L controls the amount of neighborhood information exchanged in the system. Positive L can constantly elevate testing accuracy, compared with only exchanging neighbor features in FedSage (L=0). Across different datasets, the optimal L is usually around 2. When a dataset has too many missing links between subgraphs (e.g., M=10), a large L introduces more biased deep neighbor embeddings, and thus worse performances.

4.4 Component Study of Proto We compare the downstream task performance of FedDEP under different numbers of cluster C in Proto. Table 3 shows that choosing a proper C, which controls how abstract the

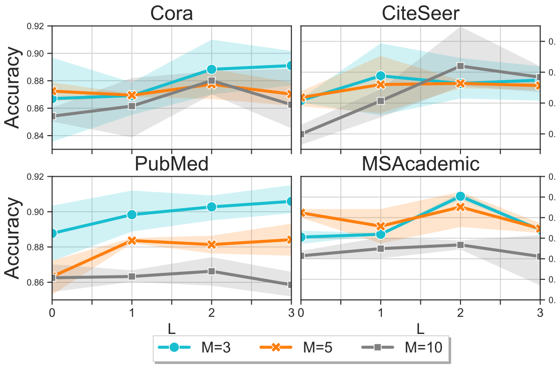

Figure 4: Component study for DGen in FedDEP with different depths L of generated neighbor embeddings on four datasets with different M's. L=0 is FedSage.

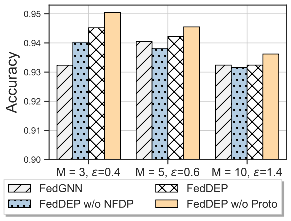

Figure 5: Component study for NFDP with different levels of edge-LDP privacy protection on MSAcademic with different client numbers.

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.

4.5 Component Study of NFDP We compare models' utility under the same DP 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. In MSAcademic dataset, when M=3, we have d=5, D=15, ε=0.4, and σ=4.2; when M=5, we have d=5, D=10, ε=0.6, and σ=2.1; when M=10, we have d=5, D=3, ε=1.4, and σ =0.4. Regarding Fig. 5, our NFDP always outperforms the noise-injected counterparts in achieving similar edge-LDP, which empirically justifies the superior utility-privacy trade-off of NFDP when compared to gradients perturbation-based approaches such as DPSGD.

4.6 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.

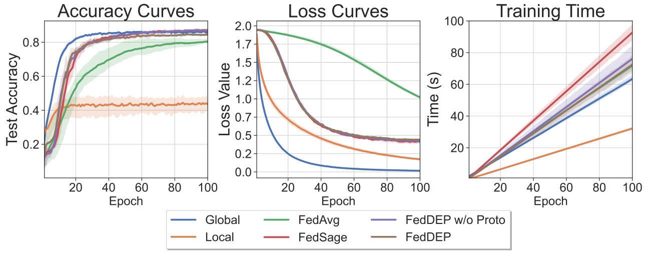

Figure 6: Training curves of different frameworks on Cora dataset with M=5. (Best viewed in color.)

5 Conclusion

In this work, we study subgraph FL by comprehensively tackling the unique challenges of utility, efficiency, and privacy. We propose FedDEP, a novel subgraph FL framework with Deep Efficient Private neighbor generation. FedDEP consists of a series of techniques including deep neighbor generation with embedding-fused graph convolution, efficient pseudo-FL with missing neighbor prototyping, and privacy guarantee with noise-free edge-LDP. Theoretical analysis and empirical results together embrace the claimed benefits of FedDEP.

Acknowledgement

This research was partially supported by 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] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *CCS*, 2016.

[2] Jinheon Baek, Wonyong Jeong, Jiongdao Jin, Jaehong Yoon, and Sung Ju Hwang. Personalized subgraph federated learning. In *ICML*, 2023.

[3] Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy profiles and amplification by subsampling. *JPC*, 10(1), 2020.

[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *JSTAT*, 2008.

[5] Fahao Chen, Peng Li, Toshiaki Miyazaki, and Celimuge Wu. Fedgraph: Federated graph learning with intelligent sampling. *IEEE TPDS*, 2021.

[6] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F Stewart, and Jimeng Sun. Gram: graph-based attention model for healthcare representation learning. In *WWW*, 2017.

[7] Woo-Seok Choi, Matthew Tomei, Jose Rodrigo Sanchez Vicarte, Pavan Kumar Hanumolu, and Rakesh Kumar. Guaranteeing local differential privacy on ultra-low-power systems. In *ISCA*, 2018.

[8] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? In *NeurIPS*, 2020.

[9] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*, 2020.

[11] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *ICML*, 2015.

[12] Thomas N Kipf and Max Welling. Variational graph auto-encoders. In *Workshop of NeurIPS*, 2016.

[13] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[14] Bill Yuchen Lin, Chaoyang He, Zihang Zeng, Hulin Wang, Yufen Huang, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. Fednlp: A research platform for federated learning in natural language processing. In *Findings of ACL: NAACL*, 2021.

[15] Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. Fedvision: An online visual object detection platform powered by federated learning. In *AAAI*, 2020.

[16] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. Feature inference attack on model predictions in vertical federated learning. In *ICDE*, 2021.

[17] Galileo Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven active surveying for collective classification. In *MLG workshop*, 2012.

[18] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *S&P*, 2019.

[19] Qiying Pan, Yifei Zhu, and Lingyang Chu. Lumos: Heterogeneity-aware federated graph learning over decentralized devices. *ICDE*, 2023.

[20] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. Generating synthetic decentralized social graphs with local differential privacy. In *SIGSAC*, 2017.

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

[22] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[23] Lichao Sun and Lingjuan Lyu. Federated model distillation with noise-free differential privacy. In *IJCAI*, 2021.

[24] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning across heterogeneous clients. In *AAAI*, 2022.

[25] Mingyue Tang, Carl Yang, and Pan Li. Graph auto-encoder via neighborhood wasserstein reconstruction. In *ICLR*, 2022.

[26] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). 2017.

[27] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. Traffic flow prediction via spatial temporal graph neural network. In *WWW*, 2020.

[28] Zhen Wang, Weirui Kuang, Yuexiang Xie, Liuyi Yao, Yaliang Li, Bolin Ding, and Jingren Zhou. Federatedscope-gnn: Towards a unified, comprehensive and efficient package for federated graph learning. In *KDD*, 2022.

[29] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Tao Qi, Yongfeng Huang, and Xing Xie. A federated graph neural network framework for privacy-preserving personalization. *Nature Communications*, 2022.

[30] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[31] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *TIST*, 10(2):1–19, 2019.

[32] Yuhang Yao, Weizhao Jin, Srivatsan Ravi, and Carlee Joe-Wong. Fedgcn: Convergence and communication tradeoffs in federated training of graph convolutional networks. *NeurIPS*, 2023.

[33] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *WWW*, 2018.

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

[35] Geng Zhao, Yi Huang, and Chi Hsu Tsai. Fedgsl: Federated graph structure learning for local subgraph augmentation. In *Big Data*, 2022.

[36] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *NeurIPS*, 2019.

[37] Qi Zhu, Carl Yang, Yidan Xu, Haonan Wang, Chao Zhang, and Jiawei Han. Transfer learning of graph neural networks with ego-graph information maximization. In *NeurIPS*, 2021.

Appendix of
Deep Efficient Private Neighbor Generation for Subgraph Federated Learning

Ke Zhang* Lichao Sun† Bolin Ding‡ Siu Ming Yiu§ Carl Yang¶

A. Proof for Embedding-fused Graph Convolution

A.1 Proof for Statement 3.1

Statement 3.1. *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.*

Proof. At k-th layer of embedding-fused graph convolution, for every node u in v's one-hop ego-graph $G^1(v)$, we denote its mean averaged node representations as $\bar{x}_u^k \in \mathbb{R}^{1\times d_h}$ and embeddings as $\bar{z}_u \in \mathbb{R}^{1\times d_z}$.

According to our description in Section 3, we have

$$x_u^k = \sigma(W^{(k)} \times [\bar{x}_u^{k-1}||\bar{z}_u]^\top)^\top,$$

where $W^{(k)} \in \mathbb{R}^{d_h\times(d_h+d_z)}$ is the learnable matrix in the convolution.

As x_u^k can also be regarded as

$$\sigma(\begin{bmatrix} W_{1,1}^{x(k)} & \cdots & W_{1,d_h}^{x(k)} & W_{1,1}^{z(k)} & \cdots & W_{1,d_z}^{z(k)} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ W_{d_h,1}^{x(k)} & \cdots & W_{d_h,d_h}^{x(k)} & W_{d_h,1}^{z(k)} & \cdots & W_{d_h,d_z}^{z(k)} \end{bmatrix} \times \begin{bmatrix} \bar{x}_{u,1}^{k-1} \\ \vdots \\ \bar{x}_{u,d_x}^{k-1} \\ \bar{z}_{u,1} \\ \vdots \\ \bar{z}_{u,d_z} \end{bmatrix})^\top,$$

which equals to $\sigma(W^{x(k)} \times \bar{x}_u^{k-1\top} + W^{z(k)} \times \bar{z}_u^\top)^\top$, where $W^{x(k)} \in \mathbb{R}^{d_h\times d_h}$ and $W^{z(k)} \in \mathbb{R}^{d_h\times d_z}$ are learnable weights in the convolution.

Therefore, we justify the correctness of embedding-fused graph convolution where the mended deep neighbors and the representations/features contribute to the convolution with respective learnable parameters, and conclude the proof. □

A.2 Proof for Statement 3.2

Lemma A.1. *For a node v, we denote the prediction, computed by one layer of embedding-fused graph convolution on its 1-hop impaired ego-graph, where every node is mended with deep neighbors computed on the respective L-hop missing context, as $\tilde{y}'_v$, and the prediction, computed by $(L+1)$ layers of graph convolution on its $(L+1)$-hop ego-graph, as $\tilde{y}_v$, where $L \in \mathbb{N}^*$. $\tilde{y}'_v$ and $\tilde{y}_v$ are the compound vectors for the same local context of v.*

Proof. For node v, we compute its prediction $\tilde{y}'_v$ as

$$\tilde{y}'_v = x_v^1 = \sigma(W^{(1)} \times [mean(\{x_u^0 | u \in G^1(v)\})||\bar{z}_v]^\top,$$

where for every $u \in G^1(v)$,

$$x_u^0 = \sigma(W^{(0)} \times [x_u||\bar{z}_u])^\top = \sigma(W^{(0)} \times [x_u||mean(z_u)])^\top$$

Since x_u^0 contains $\{x_u, z_u\}$, and $\tilde{y}'_v$ is then computed based on $\{x_u, z_u | u \in G^1(v)\} \cup \{\bar{z}_v\}$. We only need to verify $\{x_u, z_u | u \in G^1(v)\} \cup \{\bar{z}_v\}$ containing the same information as the $\{x_u | u \in G^{L+1}(v)\}$.

First we have $\{\bar{z}_v\}$ computed from the L-hop neighbors of v, *i.e.*, $\{x_u | u \in G^L(v)\}$. Then we only need to consider whether the content of $\{x_u, z_u | u \in G^1(v)\}$ covers the $\{x_u | u \in G^{L+1}(v) \backslash G^L(v)\}$. Since every $z_u^p \in z_u$ is computed on the L-hop ego-graph of node u with original graph convolution mechanism, z_u^p contains the information of $\{x_p | p \in G^L(u)\}$. Thus, the union of z_u for $u \in G^1(v)$ covers $\{x_p | p \in G^L(u), u \in G^1(v)\} = \{x_p | p \in G^{L+1}(v)\}$, which includes $\{x_u | u \in G^{L+1}(v) \setminus G^L(v)\}$.

Obviously, $\{x_u, z_u | u \in G^1(v)\} \cup \{\bar{z}_v\}$ contains the same $L+1$ ego-graph content as $\{x_u | u \in G^{L+1}(v)\}$ does, we have Lemma A.1 proved. □

Statement 3.2. *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 $\tilde{y}'_v$, and the prediction, computed by $(K+L)$ layers of graph convolution on its $(K+L)$-hop ego-graph, as $\tilde{y}_v$, where $K, L \in \mathbb{N}^*$. $\tilde{y}'_v$ and $\tilde{y}_v$ are the compound vectors for the same local context of v.*

*cszhangk@connect.hku.hk, ClusterTech Limited.
†lis221@lehigh.edu, Lehigh University.
‡bolin.ding@alibaba-inc.com, Alibaba Group.
§smyiu@cs.hku.hk, The University of Hong Kong.
¶j.carlyang@emory.edu, Emory University.

Proof. To prove Statement 3.2, we extend Lemma A.1 from 1-hop impaired ego-graph to the K-hop impaired ego-graph mended with L-hop local missing context embeddings.

By iterativly applying Lemma A.1 K-L times, we have node v's prediction $\tilde{y}'_v$ computed on $\{x_u, z_u | u \in G^K(v)\}$ with z_u containing the information of $\{x_p | p \in G^L(u)\}$. The entire content is the same as where $\tilde{y}_v$ is retrieved with original graph convolution, *i.e.*, $\{x_p | p \in G^{K+L}(u)\}$. Thus, we have Statement 3.2 proved. □

B. Proof for Theorem 3.1

LEMMA B.1. *Given a graph, with its nodes' degrees by at least D, and a GCN model for embedding computation, after one epoch of mini-batch training on 1-hop ego-graphs drawn from the graph with sampling size as d, the GCN achieves at most $(\ln \frac{D+1}{D+1-d}, \frac{d}{D})$-edge-LDP when $d < D$, and at least $(d \ln \frac{D+1}{D}, 1 - (\frac{D-1}{D})^d)$-edge-LDP otherwise.*

Proof. To prove Lemma B.1, we first revisit the NFDP mechanisms [17] on (ε, δ)-differential privacy of different sampling policies.

THEOREM B.1. (NFDP MECHANISM-I [17]) *Given a training dataset of size D, sampling without replacement achieves $(\ln \frac{D+1}{D+1-d}, \frac{d}{D})$- differential privacy, where d is the subsample size.*

THEOREM B.2. (NFDP MECHANISM-II [17]) *Given a training dataset of size D, sampling with replacement achieves $(d \ln \frac{D+1}{D}, 1 - (\frac{D-1}{D})^d)$- differential privacy, where d is the subsample size.*

To apply Theorem B.1 and Theorem B.2 in Lemma B.1, we can regard the 1-hop neighbors of the target node v, *i.e.*, the neighbors on the 1-hop ego-graph of v, as the entire dataset with size D, and the mini-batch sampling node size is the subsampling size d.

In this way, one epoch of training the GCN model with the mini-batch sampling has two cases. One case is when $d < D$, while the other is $d \geq D$. For the neighbor sampling method, we follow the implementation of FederatedScope [18], where the former case uses the sampling without replacement, and the latter case uses the sampling with replacement. Therefore, when $d < D$, the sampling can achieve $(\ln \frac{D+1}{D+1-d}, \frac{d}{D})$-differential privacy for the neighbor list, and $(d \ln \frac{D+1}{D}, 1 - (\frac{D-1}{D})^d)$-differential privacy otherwise.

To transfer the general DP to the edge-LDP, we need to analyze it according to the definition of edge-LDP and differential privacy. We revisit the definition of general DP as follows.

DEFINITION B.1. ((ε, δ)-DIFFERENTIAL PRIVACY) *A randomized mechanism $\mathcal{M} : \mathcal{A} \to B$ with domain $\mathcal{A}$ and range B satisfies (ε, δ)-differential privacy if for all two neighboring inputs $U, U' \in \mathcal{A}$ that differ by one record, and any measurable subset of outputs $S \subseteq B$ it holds that*

$$Pr[\mathcal{M}(U) \in S] \leq e^{\varepsilon} Pr[\mathcal{M}(U') \in S] + \delta \tag{.1}$$

Then we revisit the definition of edge-LDP as below.

DEFINITION B.2. *For a graph with n nodes, denote its node v's neighbor list as $(b_1, \dots, b_n)$. For $u \in [n]$, if v is linked with v, b_u is 1. Otherwise, b_u is 1. Let $\varepsilon, \delta \in \mathbb{R}_{\geq 0}$, and $R : \mathcal{G} \to \mathbb{R}$ is a randomized algorithm. R provides (ε, δ)-edge-LDP if for any two local neighbor lists γ, γ' that differ in one bit and any $S \subseteq R$,*

$$Pr[R(\gamma) \in S] \leq e^{\varepsilon} Pr[R(\gamma') \in S] + \delta. \tag{.2}$$

By regarding the input dataset U, U' in Eq. (.1) as two neighbor lists γ, γ' in Eq. (.2), we have general differential privacy transferred to edge-LDP. As the mini-batch sampling GCN can achieve γ, γ' in Eq. (.2) through whether sampling a neighbor node in the ego-graph, we transfer the sampling in NFDP of (ε, δ)-differential privacy to the equal effect of the mini-batching sampling in noise-free (ε, δ)-edge-LDP.

Since nodes on a graph can have different degrees, and the lower bound of protection implies the privacy of this mechanism, we choose the max values of (ε, δ) by calculating them using the minimum degree among all nodes. In this way, Lemma B.1 is proved. □

LEMMA B.2. *For a subgraph, given every node's L-hop ego-graph with its every L-1 hop nodes of degrees by at least D, and a GCN model for embedding computation, after N epochs of mini-batch training with each hop of sampling size as d, the GCN achieves $(\tilde{\varepsilon}, \tilde{\delta})$-edge-LDP, where*

$$\tilde{\varepsilon} = \min\{LN\varepsilon, LN\varepsilon \frac{(e^{\varepsilon} - 1)}{e^{\varepsilon} + 1} + \varepsilon U \sqrt{2LN}\},$$
$$\tilde{\delta} = (1 - \delta)^{LN} (1 - \delta'),$$

and $U = \min\{\sqrt{\ln(e + \frac{\varepsilon\sqrt{LN}}{\delta'})}, \sqrt{\ln(\frac{1}{\delta'})}\}$, *for* $\delta' \in [0, 1]$, *and* (ε, δ) *are* $(\ln \frac{D+1}{D+1-d}, \frac{d}{D})$ *and* $(d \ln \frac{D+1}{D}, 1 - (\frac{D-1}{D})^d)$ *in Lemma B.1 for respective cases.*

Proof. To prove Lemma B.2, we need to adaptively apply Lemma B.1 by N epochs on the L times of graph convolution, *i.e.*, total LN times. Thus, we revisit the Composition of DP Mechanisms [9] as follows.

THEOREM B.3. (COMPOSITION OF DP [9]) *For any $\varepsilon > 0$, $\delta, \delta' \in [0, 1] > 0$, the class of (ε, δ)-differential* *private mechanisms satisfies* $(\tilde{\varepsilon}, 1-(1-\delta)^k(1-\delta'))$*-differential private under k-fold adaptive composition, for*

$$\tilde{\varepsilon} = \min\{k\varepsilon, k\varepsilon\frac{(e^\varepsilon-1)}{e^\varepsilon+1} + \varepsilon\sqrt{2k}\min\{\sqrt{\ln(e+\frac{\varepsilon\sqrt{k}}{\delta'})}, \sqrt{\ln(\frac{1}{\delta'})}\}\}$$

By aligning general differential privacy to edge-LDP as we described in the proof of Lemma B.1, we have the same conclusion of the composition rule for edge-LDP as Theorem B.3. Then we substitute the k in the composition rule to LN, and specifying the (ϵ, δ) as the pairs in Lemma B.1. Thus, Lemma B.2 is proved. □

Theorem 3.1. (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* $(\log(1+r(e^{\tilde{\varepsilon}}\text{-}1), r\tilde{\delta})$*-edge-LDP, where*

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

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

Proof. FedDEP framework first pre-calculates the embeddings from a mini-batch trained GCN to retrieve prototype sets, then it leverages the deep neighbor generator that employs a Bernoulli sampler R with expected value r to jointly train a classifier on subgraphs mended with generated deep neighbor prototypes.

To prove Theorem 3.1, we revisit the privacy amplification by subsampling in the general DP [3].

Theorem B.4. (privacy amplification [3]) *Given a dataset U with n data records, subsampling mechanism* $\mathcal{S}$ *subsamples a subset of data* $\{d_i|\sigma_i = 1, i \in [n]\}$ *by sampling* $\sigma_i \sim Ber(p)$ *independently for* $i \in [n]$*. If mechanism* $\mathcal{M}$ *satisfied* (ε, δ)*-differential privacy, mechanism* $\mathcal{M} \circ \mathcal{S}$ *is* $(\log(1+p(e^{\varepsilon-1}), p\delta)$*-differential private.*

We prove Theorem 3.1 by applying Theorem B.4 and Lemma B.2 in four steps.

We first transfer the conclusion of Theorem B.4 into edge-LDP by following the proof of Lemma B.1. Then we specify the (ε, δ)-differential privacy mechanism $\mathcal{M}$ in Theorem B.4 as the edge-LDP embedding computation GCN model in Lemma B.2 with respective privacy-related parameters. Next, we specify the subsampling mechanism $\mathcal{S}$ in Theorem B.4 as the Bernoulli sampler in FedDEP with DGen on prototypes. By substituting the p in Theorem B.4 to r, we have Theorem 3.1 proved. □

C. Related Works

C.1 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 [5, 7, 14, 23]. Another direction is for the scenarios where every client holds a set of small graphs, such as molecular graphs for drug discovery [21]. 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 [11] or randomized DropEdge [16].

To deal with the missing neighbor problem in subgraph FL, existing works [4, 13, 19, 24–26] propose to augment local subgraphs by retrieving missing neighbors across clients, and then mend the subgraphs with the retrieved neighbor information. FedGraph [4] considers a relaxed scenario where the existences of inter-subgraph neighbors are known for corresponding clients. Lumos [13], as well as FedGraph [4], requests the central server to manage the FL process with auxiliary data. FedSage [25] primarily focuses on the design of the missing neighbor generator without considering the important aspects of efficiency and privacy. FedHG [24] 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] and FedGSL [26] equip their augmentation with privacy protection based on additional trusted authorities and/or noise injection.

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

C.2 Privacy-Preserving Learning for Graphs Privacy-preserving learning over graph data has been

Table 1: Datasets and the synthesized distributed systems statistics. $|V_i|$ and $|E_i|$ rows show the averaged numbers of nodes and links in all subgraphs, and ΔE 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
ΔE	496	552	912	178	224	247	4570	4818	9323	8995	12149	26973
$\Delta E/\|E\|$	0.0940	0.1046	0.1728	0.0391	0.0492	0.0543	0.1031	0.1087	0.2103	0.1098	0.1484	0.3294

widely studied. Differential Privacy (DP) [6] 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 [1], which injects designed noise into clipped gradients during model training. For centralized training scenarios, DPGGAN [22] incorporates DPSGD to achieve DP for individual links on original graphs. In FL systems, VFGNN [27] 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 [2,12,20], FKGE [14] and FedGSL [26] for FL systems, guarantee their proposed techniques with general DP by applying noise perturbation.

However, general DP does not depict the protections 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 [15]. 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 [15], 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. [15] propose a decentralized social graphs generation technique with the edge-LDP. Imola et al. [8] 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. [10] 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.

D. Revisit of FedSage+

In this section, we revisit the popular existing subgraph federated learning framework, i.e., FedSage+, the variant of FedSage with the proposed missing neighbor generator (NeighGen) [25]. For simplicity, in this paper, we refer to this stronger variant as FedSage.

D.1 Neighbor Generation The proposed NeighGen in [25] 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 $\tilde{n}_v$, and predicts the respective feature set $\tilde{x}_v$.

D.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 $\mathcal{L}_{i,j}^f$ between real node features on G_j and the generated ones from received data.

3. D_j sends $\mathcal{L}_{i,j}^f$'s gradients back to D_i via server S.

4. D_i computes the total gradients of cross-subgraph neighbor reconstruction loss $\mathcal{L}_i^f = \alpha^n \sum_{j\in[M]} \mathcal{L}_{i,j}^f$ by summing up all received gradients from other clients. Notably, $\mathcal{L}_{i,i}^f$ 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 [25].

E. Additional Experimental Details

We present the statistics of tested four datasets and the synthesized distributed systems in Tab. 1.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *CCS*, 2016.

[2] Faraz Ahmed, Alex X Liu, and Rong Jin. Publishing social network graph eigenspectrum with privacy guarantees. *IEEE TNSE*, 7:892–906, 2019.

[3] Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy profiles and amplification by subsampling. *JPC*, 10(1), 2020.

[4] Fahao Chen, Peng Li, Toshiaki Miyazaki, and Celimuge Wu. Fedgraph: Federated graph learning with intelligent sampling. *IEEE TPDS*, 2021.

[5] Mingyang Chen, Wen Zhang, Zonggang Yuan, Yantao Jia, and Huajun Chen. Fede: Embedding knowledge graphs in federated setting. In *IJCKG*, 2020.

[6] Cynthia Dwork. Differential privacy. In *ICALP*, 2006.

[7] Zishan Gu, Ke Zhang, Guangji Bai, Liang Chen, Liang Zhao, and Carl Yang. Dynamic activation of clients and parameters for federated learning over heterogeneous graphs. In *ICDE*, 2023.

[8] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Differentially private triangle and 4-cycle counting in the shuffle model. In *SIGSAC*, 2022.

[9] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *ICML*, 2015.

[10] Wanyu Lin, Baochun Li, and Cong Wang. Towards private learning on decentralized graphs with local differential privacy. *IEEE TIFS*, 17:2936–2946, 2022.

[11] Songtao Liu, Rex Ying, Hanze Dong, Lanqing Li, Tingyang Xu, Yu Rong, Peilin Zhao, Junzhou Huang, and Dinghao Wu. Local augmentation for graph neural networks. In *ICML*, 2022.

[12] Wentian Lu and Gerome Miklau. Exponential random graph estimation under differential privacy. In *KDD*, 2014.

[13] Qiying Pan, Yifei Zhu, and Lingyang Chu. Lumos: Heterogeneity-aware federated graph learning over decentralized devices. *ICDE*, 2023.

[14] Hao Peng, Haoran Li, Yangqiu Song, Vincent Zheng, and Jianxin Li. Differentially private federated knowledge graphs embedding. In *CIKM*, 2021.

[15] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. Generating synthetic decentralized social graphs with local differential privacy. In *SIGSAC*, 2017.

[16] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *ICLR*, 2020.

[17] Lichao Sun and Lingjuan Lyu. Federated model distillation with noise-free differential privacy. In *IJCAI*, 2021.

[18] Zhen Wang, Weirui Kuang, Yuexiang Xie, Liuyi Yao, Yaliang Li, Bolin Ding, and Jingren Zhou. Federatedscope-gnn: Towards a unified, comprehensive and efficient package for federated graph learning. In *KDD*, 2022.

[19] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Tao Qi, Yongfeng Huang, and Xing Xie. A federated graph neural network framework for privacy-preserving personalization. *Nature Communications*, 2022.

[20] Qian Xiao, Rui Chen, and Kian-Lee Tan. Differentially private network data release via structural inference. In *KDD*, 2014.

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

[22] Carl Yang, Haonan Wang, Ke Zhang, Liang Chen, and Lichao Sun. Secure deep graph generation with link differential privacy. In *IJCAI*, 2021.

[23] Kai Zhang, Yu Wang, Hongyi Wang, Lifu Huang, Carl Yang, and Lichao Sun. Efficient federated learning on knowledge graphs via privacy-preserving relation embedding aggregation. In *Findings of EMNLP*, 2022.

[24] Ke Zhang, Han Xie, Zishan Gu, Xiaoxiao Li, Lichao Sun, Siu Ming Yiu, Yuan Yao, and Carl Yang. Subgraph federated learning over heterogeneous graphs. In *FedGraph-CIKM*, 2022.

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

[26] Geng Zhao, Yi Huang, and Chi Hsu Tsai. Fedgsl: Federated graph structure learning for local subgraph augmentation. In *Big Data*, 2022.

[27] Jun Zhou, Chaochao Chen, Longfei Zheng, Xiaolin Zheng, Bingzhe Wu, Ziqi Liu, and Li Wang. Vertically federated graph neural network for privacy-preserving node classification. In *IJCAI*, 2021.