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

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

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. This work was done while at The University of Hong Kong.
†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.*, $\{x_p | p \in G^{K+L}(u)\}$. In this way, 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 [16] on (ε, δ)-differential privacy of different sampling policies.

THEOREM B.1. (NFDP MECHANISM-I [16]) *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 [16]) *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 neighbor list 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 [17], 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*

$$Pr[\mathcal{M}(U) \in S] \leq e^{\varepsilon} Pr[\mathcal{M}(U') \in S] + \delta \tag{0.1}$$

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 $(b_1, \ldots, b_n)$. 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 $S \subseteq R$,*

$$Pr[R(\gamma) \in S] \leq e^{\varepsilon} Pr[R(\gamma') \in S] + \delta. \tag{0.2}$$

By regarding the input dataset U, U' in Eq. (0.1) as two neighbor lists γ, γ' in Eq. (0.2), we have general differential privacy transferred to edge-LDP. As the mini-batch sampling GCN can achieve γ, γ' in Eq. (0.2) 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 the 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 firstly aligning general differential privacy to edge-LDP as we described in the proof of Lemma B.1, obviously, 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, 13, 22]. Another direction is for the scenarios where every client holds a set of small graphs, such as molecular graphs for drug discovery [20]. 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 [15].

To deal with the missing neighbor problem in subgraph FL, existing works [4, 18, 23, 24] propose to augment local subgraphs by retrieving missing neighbors across clients, and then mend the subgraphs with the retrieved neighbors. FedGraph [4] 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 [24] primarily focuses on the design of the missing neighbor generator without considering the important aspects of efficiency and privacy. FedHG [23] 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 [18] 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.

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

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 [21] incorporates DPSGD to achieve DP for individual links on original graphs. In FL systems, VFGNN [25] and FedGNN [18] 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, 19], and FKGE [13] 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 [14]. 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 [14], 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. [14] 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) [24]. For simplicity, in this paper, we refer to this stronger variant as FedSage.

D.1 Neighbor Generation The proposed NeighGen in [24] 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 [24].

E. Additional Experimental Details

In this section, we present the statistics of tested four datasets and the synthesized distributed systems.

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.

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

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

[14] 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.

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

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

[17] 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.

[18] 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.

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

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

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

[22] 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.

[23] 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.

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

[25] 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.