Secure Deep Graph Generation with Link Differential Privacy

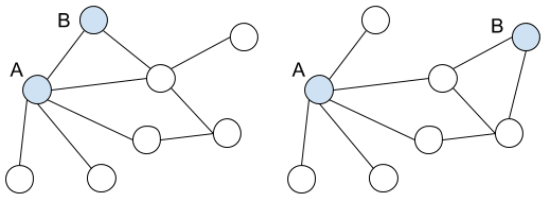
Carl Yang[1*], **Haonan Wang**[2†], **Ke Zhang**[3†], **Liang Chen**[4], **Lichao Sun**[5]
[1]Emory University
[2]University of Illinois at Urbana Champaign
[3]University of Hong Kong
[4]Sun Yat-sen University
[5]Lehigh University
j.carlyang@emory.edu, haonan3@illinois.edu, cszhangk@connect.hku.hk
chenliang6@mail.sysu.edu.cn, lis221@lehigh.edu

Abstract

Many data mining and analytical tasks rely on the abstraction of networks (graphs) to summarize relational structures among individuals (nodes). Since relational data are often sensitive, we aim to seek effective approaches to generate utility-preserved yet privacy-protected structured data. In this paper, we leverage the differential privacy (DP) framework to formulate and enforce rigorous privacy constraints on deep graph generation models, with a focus on edge-DP to guarantee individual link privacy. In particular, we enforce edge-DP by injecting proper noise to the gradients of a link reconstruction based graph generation model, while ensuring data utility by improving structure learning with structure-oriented graph discrimination. Extensive experiments on two real-world network datasets show that our proposed DPGGAN model is able to generate graphs with effectively preserved global structure and rigorously protected individual link privacy.

1 Introduction

Nowadays, open data of networks (graphs) play a pivotal role in data mining and data analytics [Yang *et al.*, 2020; Xie *et al.*, 2020]. By releasing and sharing structured relational data with research facilities and enterprise partners, data companies harvest the enormous potential value from their data, which benefits decision-making on various aspects, including social, financial, environmental, through collectively improved ads, recommendation, retention, and so on [Sigurbjörnsson and Van Zwol, 2008; Kuhn, 2009]. However, graph data usually encode sensitive information not only about individuals but also their interactions, which makes direct release and exploitation rather unsafe. More importantly, even with careful anonymization, individual privacy is still at stake under collective attack models facilitated by the underlying graph structure [Zhang *et al.*, 2019; Cai *et al.*, 2018; Sun *et al.*, 2018]. Can we find a way to securely generate graph data without drastic sanitization that essentially renders the released data useless?



(a) Anonymized original net. (b) DPGGAN generated net.

Figure 1: A toy pair of anonymized and generated networks.

In dealing with such tension between the need to release utilizable data and the concern of data owners' privacy, quite a few secure deep generative models have been proposed recently, focusing on grid-based data like images, texts and gene sequences [Papernot *et al.*, 2018; Sun *et al.*, 2020a; Sun and Lyu, 2020]. However, none of the existing models can be directly applied to the network (graph) setting. While a secure generative model on grid-based data apparently aims to preserve high-level semantics (*e.g.*, class distributions) and protect detailed training data (*e.g.*, exact images or sentences), it remains obtuse what to be preserved and what to be protected for graph data, due to its modeling of complex interactive objects.

Motivating scenario. In Figure 1, a bank aims to encourage public studies on its customers' community structures. It does so by firstly anonymizing all customers and then sharing the network (*i.e.*, (a) in Figure 1) to the public. However, an attacker interested in knowing the financial interactions (*e.g.*, money transfer) between particular customers in the bank may happen to have access to another network of a similar set of customers (*e.g.*, a malicious employee of another financial company). The similarity of simple graph properties like node degree distribution and triangle count between the two networks can then be used to identify specific customers with high accuracy in the released network (*e.g.*, customer A as the only node with degree 5 and within 1 triangle, and customer B as the only node with degree 2 and within 1 triangle). Thus, the attacker confidently knows the A and B's identities and the

*Corresponding Author
†Equal Contribution

fact that they have financial interactions in the bank, which seriously harms customers' privacy and poses potential crises.

As the first contribution in this work, we define and formulate the goals of secure graph generation as *preserving global graph structure* while *protecting individual link privacy*. Continue with the toy example, the solution we propose is to train a deep graph generation model on the original network and release the generated networks (*e.g.*, (b) in Figure 1). Towards the utility of generated networks, we require them to be similar to the original networks from a global perspective, which can be measured by various commonly graph properties (*e.g.*, network (b) has very similar degree distribution and the same triangle count as (a)). In this way, we expect many downstream data mining and analytical tasks on them to produce similar results as on the original networks. As for privacy protection, we require that information in the generated networks cannot confidently reveal the existence or absence of any individual links in the original networks (*e.g.*, the attacker may still identify customers A and B in network (b), but their link structure may have changed).

Subsequently, there are two unique challenges in learning such structure-preserved and privacy-protected graph generation models, which have never been explored so far.

Challenge 1: Rigorous protection of individual link privacy. The rich relational structures in graph data often allow attackers to recover private information through various ways of collective inference [Zhang *et al.*, 2014; Narayanan and Shmatikov, 2009]. Moreover, graph structure can always be converted to numerical features such as spectral embedding, after which most attacks on grid-based data like model inversion [Fredrikson *et al.*, 2015] and membership inference [Shokri *et al.*, 2017] can be directly applied for link identification. Existing frameworks [Nobari *et al.*, 2014; Xue *et al.*, 2012] for graph link protection only demonstrate certain types of privacy regarding specific empirical measures without principled theoretical guarantee. How can we design an effective mechanism with rigorous privacy protection on links in graphs against various attacks?

Challenge 2: Effective preservation of global graph structure. To capture the global graph structure, the model has to constantly compare the structures of the input graphs and currently generated graphs during training. However, unlike images and other grid-based data, graphs have flexible structures and arbitrary node orders. How can we allow a graph generation model to effectively learn from the structural difference between two graphs, without conducting very time-costly operations like isomorphism tests all the time?

Present work. In this work, for the first time, we draw attention to the secure generation of graph data with deep neural networks. Technically, towards the aforementioned two challenges, we develop Differentially Private Graph Generative Adversarial Networks (DPGGAN), which imposes DP training over a link reconstruction based graph generation model for rigorous individual link privacy protection, and further ensures structure-oriented graph comparison for effective global graph structure preservation. In particular, we first formulate and enforce edge-DP via gradient distortion by properly injecting designed noises during model training. Then we leverage graph convolutional networks [Kipf and Welling, 2017] through a generative adversarial network architecture [Gu *et al.*, 2019] to enable structure-oriented graph discrimination.

To evaluate the effectiveness of DPGGAN, we conduct extensive experiments on two real-world network datasets. On one hand, we evaluate the utility of generated graphs by computing a suite of commonly used graph properties to compare the global structures of generated graphs with the original ones. On the other hand, we validate the privacy of individual links by evaluating links predicted from the generated graphs on the original graphs. Consistent experimental results show that DPGGAN is able to effectively generate graphs that are similar to the original ones regarding global graph structure, while at the same time useless towards individual link prediction.

2 Related Work

Differential Privacy (DP). Differential privacy is a statistical approach in addressing the paradox of learning nothing about an individual while learning useful information about a population [Dwork *et al.*, 2014]. Recent advances in deep learning have led to the rapid development of DP-oriented learning schemes. Among them, the Gaussian Mechanism [Dwork *et al.*, 2014], defined as follows, provides a neat and compatible framework for DP analysis over machine learning models.

Definition 1 (Gaussian Mechanism [Dwork *et al.*, 2014])**.** *For a deterministic function f with its ℓ_2-norm sensitivity as $\Delta_2 f = \max_{\|\mathbf{G}-\mathbf{G}'\|_1=1} \|f(\mathbf{G}) - f(\mathbf{G}')\|_2$, we have:*

$$\mathcal{M}_f(\mathbf{G}) \triangleq f(\mathbf{G}) + \mathcal{N}(0, \Delta_2 f^2 \sigma^2), \tag{1}$$

where $\mathcal{N}(0, \Delta_2 f^2 \sigma^2)$ is a random variable obeying the Gaussian distribution with mean 0 and standard deviation $\Delta_2 f \sigma$. The randomized mechanism $\mathcal{M}_f(\mathbf{G})$ is (ε, δ)-DP if $\sigma \geq \Delta_2 f \sqrt{2\ln(1.25/\delta)}/\varepsilon$ and $\varepsilon < 1$.

Following this framework, [Abadi *et al.*, 2016] proposes a general training strategy called DPSGD, which looses the condition on the overall privacy loss than that in Definition 1 by tracking detailed information of the SGD process to achieve an adaptive Gaussian Mechanism.

DP learning has also been widely adapted to generative models [Frigerio *et al.*, 2019; Papernot *et al.*, 2018; Sun *et al.*, 2020b]. For example, [Frigerio *et al.*, 2019] enforces DP on the discriminators, and thus inductively on the generators, in a GAN scheme. However, none of them can be directly applied to graph data due to the lack of consideration of structure generation.

For graph structured data, two types of privacy constraints can be applied, *i.e.*, node-DP [Kasiviswanathan *et al.*, 2013] and edge-DP [Blocki *et al.*, 2012], which define two neighboring graphs to differ by at most one node or edge. In this work, we aim at secure graph generation, and particularly, we focus on edge privacy because it is essential for the protection of object interactions unique for graph data. Several existing works have studied the protection of edge-DP. For example, [Sala *et al.*, 2011] generates graphs based on the statistical representations extracted from the original graphs blurred by designed noise, whereas [Wang and Wu, 2013] enforces the

parameters of dK-graph models to be private. However, based on shallow graph generation models, they do not flexibly capture global graph structure that can support various unknown downstream analytical tasks.

Graph Generation (GGen). GGen has been studied for decades and is widely used to synthesize network data for developing various collective analysis and mining models. Earlier works mainly use probabilistic models to generate graphs with certain properties [Watts and Strogatz, 1998; Barabási and Albert, 1999], which are manually designed based on sheer observations and prior assumptions.

Thanks to the surge of deep learning, many advanced GGen models have been developed recently, which leverage different powerful neural networks in a learn-to-generate manner [Kipf and Welling, 2016; Bojchevski *et al.*, 2018; You *et al.*, 2018; Simonovsky and Komodakis, 2018] For example, NetGAN [Bojchevski *et al.*, 2018] converts graphs into biased random walks, learns the generation of walks with GAN, and assembles the generated walks into graphs; GraphRNN [You *et al.*, 2018] regards the generation of graphs as node-and-edge addition sequences, and models it with a heuristic breadth-first-search scheme and hierarchical RNN. These deep learning models can often generate graphs with much richer properties, and flexible structures learned from real-world networks.

To the best of our knowledge, no existing work on deep GGen has looked into the potential privacy threats laid during the learning and generation with powerful models. Such concerns are rather urgent in the graph setting, where sensitive information can often be more easily compromised in a collective manner [Zhang *et al.*, 2014], and privacy leakage can easily further propagate [Sun *et al.*, 2018].

3 DPGGAN

In this work, we propose DPGGAN for securely generating graphs, whose global structures are similar to the original sensitive ones, but the individual links (edges) between objects (nodes) are safely protected. To provide robust privacy guarantees towards various graph attacks, we propose to leverage the well-studied technique of differential privacy (DP) [Dwork *et al.*, 2014] by enforcing the edge-DP defined as follows.

Definition 2 (Edge Differential Privacy [Blocki *et al.*, 2012])**.** *A randomized mechanism $\mathcal{M}$ satisfies (ε, δ)-edge-DP if for any two neighboring graphs $\mathbf{G}_1, \mathbf{G}_2 \in \mathcal{G}$, which differ by at most one edge, $\Pr[\mathcal{M}(\mathbf{G}_1) \in S] \leq exp(\varepsilon) \times \Pr[\mathcal{M}(\mathbf{G}_2) \in S] + \delta$, where $S \subset range(\mathcal{M})$.*

Our key insight is, *a graph generation model $\mathcal{M}$ satisfying the above edge-DP should learn to generate similar graphs given the input of two neighboring graphs that differ by at most one edge; as a consequence, the information in the generated graph does not confidently reveal the existence or absence of any one particular edge in the original graph, thus rigorously protecting individual link privacy.*

To ensure DP on individual links, we exploit the existing link reconstruction based graph generation model GVAE [Kipf and Welling, 2016], and design a training algorithm to dynamically distort the gradients of its sensitive model parameters by injecting proper amounts of Gaussian noise based on the framework of DPSGD [Abadi *et al.*, 2016]. We provide theoretical analysis on applying DPSGD to achieve edge-DP with GVAE based on the nature of graph data and the link reconstruction loss. Moreover, to improve the capturing of global graph structures, we replace the direct binary cross-entropy (BCE) loss on graph adjacency matrices in GVAE with a structure-oriented graph discriminator based on GCN and the framework of VAEGAN [Gu *et al.*, 2019]. We further prove the improved model to maintain the same edge-DP.

Backbone GVAE. Recent research on graph models has been primarily focused around GCN [Kipf and Welling, 2017], which is shown to be promising in calculating universal graph representations [Maron *et al.*, 2019; Xu *et al.*, 2019]. To guarantee individual link privacy without severely damaging global network structure, in this work, we harness the power and simplicity of GCN under the consideration of edge-DP by adapting the link reconstruction based graph variational autoencoder (GVAE) [Kipf and Welling, 2016] as our backbone graph generation model.

Notably, we are given a graph $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$, where $\mathbf{V}$ is the set of N nodes (vertices), and $\mathbf{E}$ is the set of M links (edges), which can be further modeled by a binary adjacency matrix $\mathbf{A}$. As a common practice [Hamilton *et al.*, 2017], we set the node features $\mathbf{X}$ simply as the one-hot node identity matrix. The autoencoder architecture of GVAE consists of a GCN-based graph encoder to guide the learning of a feedforward neural network (FNN) based adjacency matrix decoder, which can be trained to directly reconstruct graphs with similar links as in the input graphs. A stochastic latent variable $\mathbf{Z}$ is further introduced as the latent representation of $\mathbf{A}$ as

$$q(\mathbf{Z}|\mathbf{X}, \mathbf{A}) = \prod_{i=1}^{N} q(\mathbf{Z}_i|\mathbf{X}, \mathbf{A}) = \prod_{i}^{N} \mathcal{N}(\mathbf{z}_i|\mu_i, \text{diag}(\sigma_i^2)), \tag{2}$$

where $\mu = \mathbf{g}_\mu(\mathbf{X}, \mathbf{A})$ is the matrix of mean vectors μ_i, and $\sigma = \mathbf{g}_\sigma(\mathbf{X}, \mathbf{A})$ is the matrix of standard deviation vectors σ_i. $\mathbf{g}_\bullet(\mathbf{X}, \mathbf{A}) = \tilde{\mathbf{A}}\text{ReLU}(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_0)\mathbf{W}_1$ is a two-layer GCN model. $\mathbf{g}_\mu$ and $\mathbf{g}_\sigma$ share the first-layer parameters $\mathbf{W}_0$. $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix of $\mathbf{G}$, with degree matrix $\mathbf{D}_{ii} = \sum_{j=1}^{N} \mathbf{A}_{ij}$. $\mathbf{g}_\mu$ and $\mathbf{g}_\sigma$ form the encoder network.

To generate a graph $\mathbf{G}'$, a reconstructed adjacency matrix $\mathbf{A}'$ is computed from $\mathbf{Z}$ by an FNN decoder

$$p(\mathbf{A}|\mathbf{Z}) = \prod_{i=1}^{N}\prod_{j=1}^{N} p(\mathbf{A}_{ij}|\mathbf{z}_i, \mathbf{z}_j) = \prod_{i=1}^{N}\prod_{j=1}^{N} \sigma(\mathbf{f}(\mathbf{z}_i)^T\mathbf{f}(\mathbf{z}_j)), \tag{3}$$

where $\sigma(z) = 1/(1+e^{-z})$, $\mathbf{f}$ is a two-layer FNN appended to $\mathbf{Z}$ before the logistic sigmoid function. The whole model is trained through optimizing the following variational lower bound

$$\begin{aligned}\mathcal{L}_{vae} &= \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{prior}} \\ &= \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{A}|\mathbf{Z})] - D_{\text{KL}}(q(\mathbf{Z}|\mathbf{X}, \mathbf{A})\|p(\mathbf{Z})),\end{aligned} \tag{4}$$

where $\mathcal{L}_{\text{rec}}$ is implemented as the sum of an element-wise binary cross entropy (BCE) loss between the adjacency matrices of the input and generated graphs, and $\mathcal{L}_{\text{prior}}$ is a prior

loss based on the Kullback-Leibler divergence towards the Gaussian prior $p(\mathbf{Z}) = \prod_{i=1}^{N} p(\mathbf{z}_i) = \prod_{i}^{N} \mathcal{N}(\mathbf{z}_i|\mathbf{0}, \mathbf{I})$.

Enforcing DP. The probabilistic nature of $\mathbf{Z}$ allows the model to be generative, meaning that after training the model with an input graph $\mathbf{G}$, we can detach and disregard the encoder, and then freely generate an unlimited amount of graphs $\mathbf{G}'$ with similar links to $\mathbf{G}$, by solely drawing random samples of $\mathbf{Z}$ from the prior distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and computing $\mathbf{A}'$ with the learned decoder network *w.r.t.* Eq. (3). However, as shown in [Kurakin *et al.*, 2017], powerful neural network models like VAE can easily overfit training data, so directly releasing a trained GVAE model poses potential privacy threats.

In this work, we care about the generation model's rigorously protecting the privacy of individual links in the training data, *i.e.*, ensuring edge-DP. Particularly, in Definition 2, the inequality guarantees that the distinguishability of any one edge in the graph will be restricted to the privacy leak level proportional to ε, quantifying the absolute value of privacy information possibly to be leaked by a graph generation model.

According to Eq. (3), GVAE essentially takes a graph $\mathbf{G}$, in particular, the links $\mathbf{E}$ among the nodes $\mathbf{V}$ in $\mathbf{G}$, as input and generates a new graph $\mathbf{G}'$ by reconstructing the links $\mathbf{E}'$ among the same set of nodes $\mathbf{V}$. Therefore, if we regard GVAE as the mechanism $\mathcal{M}$, as long as its model parameters are properly randomized, the framework satisfies edge-DP. To be specific, any two input graphs $\mathbf{G}_1$ and $\mathbf{G}_2$ differing by at most one link in principle lead to similar generated graphs $\mathbf{G}'$, so information in $\mathbf{G}'$ does not confidently reveal the existence or absence of any particular link in $\mathbf{G}_1$ or $\mathbf{G}_2$.

To exploit the well-structured graph generation framework of GVAE, we leverage the Gaussian mechanism (Definition 1) [Dwork *et al.*, 2014] and DPSGD [Abadi *et al.*, 2016] to enforce edge-DP on it. In our setting, $\mathbf{G}$ is the original training graph. Then Eq. (1) tells us that a link reconstruction based graph generation model $\mathcal{M}$ can be randomized to ensure (ε, δ)-edge-DP with properly parameterized Gaussian noise. Prominently, we follow DPSGD [Abadi *et al.*, 2016] to inject a designed Gaussian noise to the gradients of our decoder network clipped by a hyper-parameter C as follows

$$\tilde{g}_{\theta,\mathcal{L}} = \frac{1}{N}\left(\sum_{i=1}^{N}\left(\nabla_{v_i,\theta}\mathcal{L}/\max(1, \frac{\|\nabla_{v_i,\theta}\mathcal{L}\|_2}{C})\right) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right), \tag{5}$$

where $\mathcal{L}$ is the loss function of a link reconstruction based graph generation model, C is the clipping hyper-parameter for the model's original gradient to bound the influence of each link, and σ is the noise scale hyper-parameter.

Now we prove that the noised clipped gradient $\tilde{g}_{\theta,\mathcal{L}}$ applied as above guarantees the learned graph generation model to be edge-DP, with a different condition from that in Definition 1 due to the nature of graph generation.

Theorem 1. *In training a link reconstruction based graph generation model on a graph with N nodes with batch size B, given the sampling probability $q = B/N$, and the number of steps T, there exist explicit constants c_1 and c_2 that for any $\varepsilon < c_1 q^2 T$, iteratively updating the model T times with $\tilde{g}_{\theta,\mathcal{L}}$ attains it with (ε, δ)-edge-DP for any $\delta > 0$ if we choose*

$$\sigma \geq c_2 \frac{q\sqrt{T\log(1/\delta)}}{\varepsilon},$$

where $c_1 \geq \frac{1}{c_0}\log\frac{1}{q\sigma}$, $c_2 \leq 1/\sqrt{c_0(1-c_0)}$ for any $c_0 \in (0, 1)$.

The proofs of Theorem 1 are detailed in Appendix A.

For the training of the DPGVAE decoder, $\mathcal{L}$ in Eq. (5) is specified as $\mathcal{L}_{rec}$ in Eq. (4). Due to the link reconstruction nature of DPGVAE, we derive the following Corollary.

Corollary 1.1 (DPGVAE edge-DP)**.** *Under the same conditions in Theorem 1, iteratively updating the decoder in* DPGVAE *for T times with $\tilde{g}_{\theta,\mathcal{L}_{rec}}$ attains it with (ε, δ)-edge-DP.*

In the generation stage, we can disregard the encoder and only use the decoder to generate an unlimited amount of graphs from randomly sampled vectors from the prior distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Due to the randomness of the normal Gaussian distribution, the sampling process can be regarded as $(0, 0)$-DP. By the composability property of DP [Dwork *et al.*, 2014], generating graphs from random noises with the DPGVAE decoder satisfies (ε, δ)-edge-DP, whose release in principle does not disclose sensitive information regarding individual links in the original sensitive networks. Since we do not release the encoder network, we do not need to clip and perturb its gradients during training to induce minimum interruptions.

Improving structure learning. Besides individual link privacy, we also aim to preserve the global network structure to ensure the utility of released data. As we discuss before, original GVAE computes the reconstruction loss between input and generated graphs based on the element-wise BCE between their adjacency matrices. Such a computation is specified on each link, rather than the graph structure as a whole. To improve the global graph structure learning, we leverage GCN again, which has been shown universally powerful in capturing graph-level structures [Maron *et al.*, 2019; Xu *et al.*, 2019]. Therefore, we borrow the framework of VAE-GAN and compute a structure-oriented generative adversarial network (GAN) loss as

$$\begin{aligned}\mathcal{L}_{gan} &= \log(\mathcal{D}(\mathbf{A})) + \log(1 - \mathcal{D}(\mathbf{A}')) \\ &\text{with } \mathcal{D}(\mathbf{A}) = \mathbf{f}'(\mathbf{g}'(\mathbf{X}, \mathbf{A})),\end{aligned} \tag{6}$$

where $\mathbf{g}'$ and $\mathbf{f}'$ are GCN and FNN networks similar as defined before, besides at the end of $\mathbf{g}'$ the node-level representations are summed up as the graph-level representation, which resembles the recently proposed GIN model for graph-level representation learning [Xu *et al.*, 2019]. In this DPGGAN framework, the decoder also serves as the generator, while $\mathcal{D} = \mathbf{f}' \cdot \mathbf{g}'$ is the discriminator.

Following [Gu *et al.*, 2019], the encoder is trained *w.r.t.* $\mathcal{L}_{rec} + \lambda_1 \mathcal{L}_{prior}$, the generator *w.r.t.* $\mathcal{L}_{rec} - \lambda_2 \mathcal{L}_{gan}$, and the discriminator *w.r.t.* $\lambda_2 \mathcal{L}_{gan}$, where λ_1 and λ_2 are hyper-parameters. In practice, to apply DPSGD to our graph generation model, we consider various options for noise injection towards the appropriate DP constraints, such as adding noises to the GCN encoder, latent graph representation, FNN decoder, GCN discriminator or any of their combinations. With theoretical justification and empirical analysis, we find injecting a designed Gaussian noise to the clipped gradients of our decoder network sufficiently leads to edge-DP of generated graphs while best preserving the desired global graph structures. Therefore, Eq. (5) with $\mathcal{L}_{rec}$ substituted by

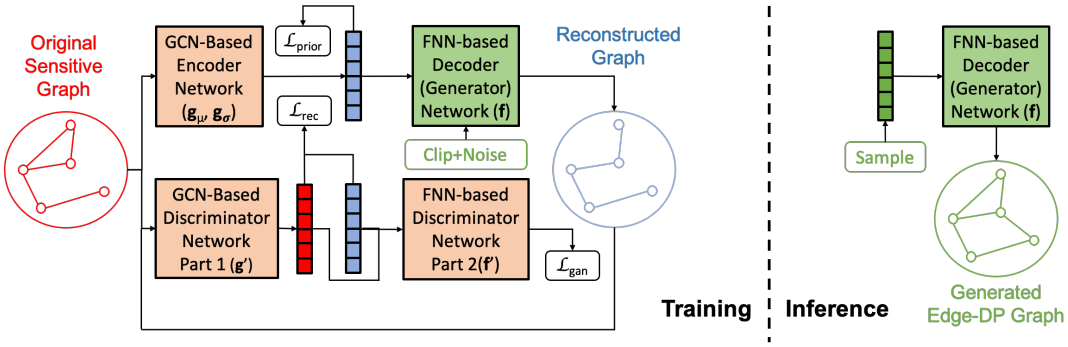


Figure 2: Neural architecture of DPGGAN (best viewed in color): Our novel graph generation model consists of a GCN-based encoder, an FNN-based decoder (generator), and a GCN+FNN-based discriminator. Sensitive data and modules are marked as red, while safe operations (*i.e.*, gradient clipping, noise injection and sampling) are marked as green, leading to DP modules and data.

$\mathcal{L}_{rec} - \lambda_2 \mathcal{L}_{gan}$ is applied to distort the gradients of the decoder (generator) and guarantee edge-DP, which can be used to securely generate networks with the other parts disregarded after training. Furthermore, to achieve the best performance without losing the DP guarantee, we gradually reduce C in Eq. (5) for an adaptive perturbation.

The overall framework. The overall framework of DPGGAN is shown in Figure 2, and the training process is detailed in Algorithm 1.

In Figure 2, the original graph is fed into the GCN-based encoder network $\mathbf{g}$ to compute node embeddings, which is then compared with the prior distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and fed into the FNN-based decoder/generator network $\mathbf{f}$ to produce the reconstructed graph and generated graph. After going through the GCN-based discriminator network part 1 ($\mathbf{g}'$), a reconstruction loss is computed between the reconstructed graph and the original graph, and a discrimination loss is computed for the generated graph and original graph after the FNN-based discriminator network part 2 ($\mathbf{f}'$).

In Algorithm 1. For a better description, we shorten $g_{\theta,(\mathcal{L}_{rec}-\lambda_2\mathcal{L}_{gan})}$ ($\tilde{g}_{\theta,(\mathcal{L}_{rec}-\lambda_2\mathcal{L}_{gan})}$) as g_θ ($\tilde{g}_\theta$). In the algorithm, for proper gradient distortion, we devise gradient clipping in Line 15 and noise injection in Line 16, which is only applied to the generator network $\mathbf{f}$ in Line 19. In Line 22, we gradually reduce the clipping hyper-parameter C as the volume of gradients decreases along training.

In the experimental analysis, existing works often fix δ as a dataset-specific value like 10^{-5}, and then analyze the performance of models based on fixed privacy budget ε [Abadi *et al.*, 2016] or fixed noise scale σ [Papernot *et al.*, 2018]. According to Theorem 1, our experiments are conducted with fixing the noise scale $\sigma = 2q\sqrt{T\log(1/\delta)}/\varepsilon$, where $\varepsilon < 2\log\frac{1}{q\sigma}q^2T$. In this work, other σ, to control the variance, we also fix δ, the sampling ratio q, for better analysis of the model's privacy loss. Note that we vary the number of training iterations (query number) T to study the relation between the model's performance and the privacy budget.

The intuition behind the novel design of DPGGAN is, the GCN encodings $\mathbf{g}'(\mathbf{A})$ and $\mathbf{g}'(\mathbf{A}')$ capture the graph structures of $\mathbf{G}$ and $\mathbf{G}'$, so a reconstruction loss $\mathcal{L}_{rec} = \|\mathbf{g}'(\mathbf{A}) - \mathbf{g}'(\mathbf{A}')\|_2^2$ captures the intrinsic structural difference between $\mathbf{G}$ and $\mathbf{G}'$ instead of the simple sum of the differences over their individual links. Note that the effectiveness of our structure-oriented discriminator is critical not only because it can directly enforce effective training of the graph generator through the minimax game in Eq. (6), but also because it can learn to relax the penalty on certain individual links through flexible and diverse configurations of the whole graph as long as the global structures remain similar, which exactly fulfills our goals of secure network release. The benefits of such diversity enabled by the VAEGAN have also been discussed in image generation [Gu *et al.*, 2019].

Algorithm 1 DPGGAN

Input : Graph data $\mathbf{G}(\mathbf{A}, \mathbf{X})$, clipping parameter C, decay ratio γ, privacy budget ε, noise scale σ, total number of nodes N, batch size $B = qN$, learning rate η, maximum number of training epochs T, loss weighing parameters λ_1 and λ_2

Output : Differentially private decoder $\mathbf{f}$.

1 Initialize weights randomly for $\mathbf{g}_\mu$, $\mathbf{g}_\sigma$, $\mathbf{f}$, $\mathbf{g}'$ and $\mathbf{f}'$.
2 **for** *epoch* $t = 0$ *to* T **do**
3 **for** *iteration* $i = 0$ *to* N/B **do**
4 Sample a subgraph $\mathbf{G}_{sub}(\mathbf{A}_{sub}, \mathbf{X}_{sub})$ of size B
5 Mean vector: $\mu \leftarrow \mathbf{g}_\mu(\mathbf{X}_{sub}, \mathbf{A}_{sub})$
6 Standard deviation vector: $\sigma \leftarrow \mathbf{g}_\sigma(\mathbf{X}_{sub}, \mathbf{A}_{sub})$
7 Update $q(\mathbf{Z}|\mathbf{X}, \mathbf{A}) \leftarrow \prod_{i=1}^{N} \mathcal{N}(\mathbf{z}_i|\mu_i, \mathrm{diag}(\sigma_i^2))$
8 Sample $\mathbf{z}_i, \mathbf{z}_j \sim q(\mathbf{Z}|\mathbf{X}, \mathbf{A})$
9 Reconstruct adjacent matrix $\mathbf{A}' \leftarrow \sigma(\mathbf{f}(\mathbf{z}_i)^T, \mathbf{f}(\mathbf{z}_j))$
10 $\mathcal{L}_{\text{prior}} = D_{\text{KL}}(q(\mathbf{Z}|\mathbf{X}, \mathbf{A})\|p(\mathbf{Z}))$
11 $\mathcal{L}_{gan} = \log(\mathcal{D}(\mathbf{A})) + \log(1 - \mathcal{D}(\mathbf{A}'))$
12 **for** *node* $x_i \in \mathbf{G}_{sub}$ **do**
13 Compute $g_\theta(x_i) \leftarrow \partial(\mathcal{L}_{rec} - \lambda_2\mathcal{L}_{gan})/\partial x_i$
14 **end**
15 Clip gradient: $\bar{g}_\theta(x_i) \leftarrow g_\theta(x_i)/max(1, \frac{\|g_\theta(x_i)\|_2}{C})$
16 Perturb gradient $\tilde{g}_\theta \leftarrow \frac{1}{B}(\sum_i \bar{g}_\theta(x_i) + \mathbf{N}(\mathbf{0}, \sigma^2C^2\mathbf{I}))$
17 Average gradient: $g_\theta \leftarrow \frac{1}{B}\sum_i g_\theta(x_i)$
18 //apply DP learning to the generator
19 Update $\mathbf{f} \overset{+}{\leftarrow} \eta \cdot \tilde{g}_\theta$; $\mathbf{f}', \mathbf{g}' \overset{+}{\leftarrow} \eta \cdot \nabla_{\mathbf{g}',\mathbf{f}'}\lambda_2\mathcal{L}_{gan}$;
20 Update $\mathbf{g}_\mu, \mathbf{g}_\sigma \overset{+}{\leftarrow} \eta \cdot \nabla_{\mathbf{g}}(\mathcal{L}_{rec} + \lambda_1\mathcal{L}_{prior})$
21 **end**
22 Update $\mathbf{C} = \mathbf{C} * \gamma$
23 **end**

Compared with DPGVAE, DPGGAN does not directly compute the link reconstruction loss based on BCE in Eq. (4), but rather computes it based on the graph discriminator $\mathcal{D}$. However, the link reconstruction based graph generator of DPGGAN is exactly the same as DPGVAE. Since we also do not release $\mathcal{D}$ after training, we can simply retrieve Corollary 1.2 from Theorem 1 as follows.

Corollary 1.2 (DPGGAN edge-DP)**.** *Under the same conditions in Theorem 1, iteratively updating the generator in* DPGGAN *for T times with $\tilde{g}_{\theta,(\mathcal{L}_{rec}-\lambda_2\mathcal{L}_{gan})}$ attains it with (ε, δ)-edge-DP.*

With Corollary 1.2, we attain DPGGAN with the same (ε, δ)-edge-DP protection of DPGVAE. For both DPGVAE and DPGGAN, the decoder/generator networks only get exposed to the noised and clipped gradients, representing the partial sensitive information within the training graphs. Hence, it prevents the inference of training graphs from both learned model parameters and generated graphs.

4 Experimental Evaluations

We conduct two sets of experiments to evaluate the effectiveness of DPGGAN in *preserving global network structure* and *protecting individual link privacy*. All code and data are in the Supplementary Materials accompanying the submission.

Experimental settings. To provide a side-to-side comparison between the original networks and generated networks, we use two standard datasets of real-world networks, *i.e.*, DBLP, and IMDB. DBLP includes 72 networks of author nodes and co-author links, where the average numbers of nodes and links are 177.2 and 258; IMDB includes 1500 networks of actor/actress nodes and co-star links, with average node and link numbers 13 and 65.9. To facilitate a better understanding towards how the graph statistics reflect the global network structure captured by the models, we also provide results of two recent deep network generation methods, *i.e.*, NetGAN [Bojchevski *et al.*, 2018] and GraphRNN [You *et al.*, 2018], with default hyper-parameter settings and no DP constraints.

For GVAE and our models, we use two-layer GCNs with sizes $32 \rightarrow 16$ for both $\mathbf{g}_\mu$ and $\mathbf{g}_\sigma$ of the encoder network, where the first layer is shared. We use two-layer FNNs with sizes $16 \rightarrow 32$ for $\mathbf{f}$ of the decoder (generator) network. For DPGGAN, we use another two-layer GCN with the same sizes for $\mathbf{g}'$ and a three-layer FNN with sizes $16 \rightarrow 32 \rightarrow 1$ for $\mathbf{f}'$. For DP-related hyper-parameters, we follow existing works [Abadi *et al.*, 2016; Shokri and Shmatikov, 2015] to fix δ to 10^{-5}, σ to 5, and q to 0.01 (which determines the batch size B as $B = qN$ with N as the graph size). Then we vary ε from 0.1 to 10 to see how much graph-level utilities are preserved under different privacy budgets. According to Eq. (6), we terminate the training of DPGGAN at T iterations when ε is depleted. Other than the essential hyper-parameters in Eq. (6), we empirically set the clipping hyper-parameter C to 5, decay ratio γ to 0.99, learning rate η to 10^{-3}, and the loss weighing hyper-parameters λ_1 and λ_2 both to 0.1. We do not observe the model to be very sensitive to the setting of these non-essential hyper-parameters.

All experiments are done with four GeForce GTX 1080 GPUs and a 12-core 2.2GHz CPU. The training time of DP-enforced models is often slightly shorter due to early stops when the privacy budget runs out, (*e.g.*, a typical train of GVAE, DPGVAE, and DPGGAN takes 60, 42 and 53 seconds on average on DBLP). The generation times of the three models are roughly the same (*e.g.*, 0.02 second on average on DBLP). As a direct comparison, NetGAN and GraphRNN take longer times under the same settings, especially for the generation (*e.g.*, 89, and 4.5 seconds for NetGAN to train and generate, and 75 and 2.4 seconds for GraphRNN, on DBLP). Although efficiency is not our primary concern, short runtimes (especially for generation) are favorable for efficient data share.

Preserving global structures. To show that DPGGAN effectively captures global network structures, we compare it and DPGVAE under different privacy budgets (controlled by ε in Eq. (6)), regarding a suite of graph statistics commonly used to evaluate the performance of graph generation models, especially from a global perspective [Bojchevski *et al.*, 2018; You *et al.*, 2018; Yang *et al.*, 2019]. [1] In particular, we train all models from scratch to convergence on each graph in the datasets. Each time, the trained model is used to generate one graph, which is compared with the original one regarding the suite of graph statistics. Then we average the absolute differences between generated and original graphs, ensuring that the positive and negative differences do not cancel out.

Beyond the single value statistics, we also compare the generated graph regarding degree distribution and motif counts. For degree distribution, we convert each graph into a 50-dim vector (all nodes with degree larger than 50 are binned together); for motif counts, we enumerate all 29 undirected motifs with 3-5 nodes and convert each graph into a 29-dim vector by motif matching. We compute the average cosine similarity between pairs of original graphs and generated graphs. Furthermore, we use graph classification, the most widely studied graph-level downstream task, to evaluate the global utilities of generated graphs. Particularly, we evaluate the accuracy of GIN [Xu *et al.*, 2019], the state-of-the-art graph classification model, with the default hyper-parameter setting.

In Table 1, our DP-constrained models constantly yield highly competitive and even better results compared with the non-private baselines of NetGAN and GraphRNN regarding global graph structural similarity between generated and original networks on both datasets. As we gradually increase the privacy budget ε, our both models apparently perform better, showing the effectiveness of our privacy constraints and a clear trade-off between privacy and utility. DPGGAN consistently outperforms DPGVAE under the same privacy budgets, supporting our novel design of the GAN framework. Moreover, as shown in Table 2, the graphs generated by DPGGAN are competitively similar to the original graphs regarding both degree distributions and motif counts, while achieving satisfactory graph classification accuracy. All these results demonstrate the global structure preservation ability of DPGGAN.

[1] Statistics we use including LCC (size of the largest connected component), TC (triangle count), CPL (characteristic path length), GINI (gini index) and REDE (relative edge distribution entropy).

	DBLP Networks					IMDB Networks				
Models	**LCC**	**TC**	**CPL**	**GINI**	**REDE**	**LCC**	**TC**	**CPL**	**GINI**	**REDE**
Original	107.5	59.90	3.6943	0.3248	0.9385	13.001	305.9	1.2275	0.1222	0.9894
GVAE (no DP)	7.51	66.93	0.1330	0.0213	0.0084	0.0145	25.83	0.0121	0.0030	0.0016
GGAN (no DP)	7.23	56.29	0.1293	0.0201	0.0057	0.0040	21.71	0.0109	0.0010	0.0012
NetGAN (no DP)	9.66	39.87	0.1943	0.0105	0.0022	0.0083	27.54	0.0192	0.0042	0.0011
GraphRNN (no DP)	10.27	57.43	0.2043	0.0415	0.0052	0.0594	27.26	0.0214	0.0155	0.0094
DPGVAE(ε=10)	**21.96**	175.29	**0.2471**	**0.0339**	**0.0153**	0.0147	43.63	0.0367	0.0036	0.0030
DPGVAE(ε=1)	23.80	187.20	0.3059	**0.0343**	0.0156	0.0253	43.73	0.0373	0.0038	0.0031
DPGVAE(ε=0.1)	26.07	215.13	0.3342	0.0344	0.0158	0.0320	44.12	0.0392	0.0042	0.0032
DPGGAN(ε=10)	**9.24**	**64.75**	**0.2035**	**0.0224**	**0.0093**	**0.0040**	**22.89**	**0.0164**	**0.0010**	**0.0017**
DPGGAN(ε=1)	**12.38**	**70.97**	**0.2643**	0.0353	**0.0117**	**0.0053**	**23.81**	**0.0168**	**0.0029**	**0.0023**
DPGGAN(ε=0.1)	24.62	**77.41**	0.2713	0.0485	0.0191	**0.0113**	**24.91**	**0.0168**	**0.0029**	**0.0025**

Table 1: Performance evaluation over compared models regarding a suite of important graph structural statistics. The Original rows include the values of original networks, while the rest rows are the average absolute difference between generated networks by different models and the original networks. Therefore, *smaller values* indicate better capturing of global network structure and thus *better global data utility*. Bold font is used to highlight the top-3 models with edge-DP constraints.

	DBLP Networks			IMDB Networks		
Models	**Degree dist.**	**Motif ct.**	**GIN acc.**	**Degree dist.**	**Motif ct.**	**GIN acc.**
GVAE (no DP)	**0.6171**	0.4093	0.3029	**0.5132**	**0.4129**	**0.4698**
GGAN (no DP)	**0.6258**	**0.4231**	**0.3374**	**0.5493**	**0.4279**	**0.4800**
NetGAN (no DP)	0.5754	**0.4109**	**0.3471**	0.4921	0.3891	0.4350
GraphRNN (no DP)	0.5454	0.3672	0.3210	0.4635	0.3721	0.3875
DPGVAE(ε=1)	0.5476	0.4038	0.3043	0.5081	0.4021	0.4625
DPGGAN(ε=1)	**0.6092**	**0.4150**	**0.3261**	**0.5486**	**0.4150**	**0.4725**

Table 2: Performance evaluation regarding degree distribution, motif counts and GIN accuracy. *Larger values* for both cosine similarity and GIN accuracy indicate *better graph utility*. Note that the GIN model trained on the original networks achieves 0.3578 and 0.5163 accuracy on the two datasets, which provides the upper bounds for all compared graph generation models. Bold font is used for values ranked top-3.

Protecting individual links. To show that DPGGAN effectively guarantees individual link privacy, we train models shown in Figure 3 for another K times on each dataset. Instead of complete networks, we randomly sample 80% of the original networks' links to train the models. After training and generation, we use degree distribution to align the nodes in the generated networks with those in the original networks. Then we evaluate the individual link prediction accuracy by comparing links predicted in the generated networks and links hidden during training in the original networks.

As shown in Figure 3, for both datasets, links predicted on the networks generated by DPGGAN models are much *less accurate* than those predicted on the original networks (13.0%-16.9% and 27.3%-28.6% accuracy drops on DBLP and IMDB, respectively) as well as the networks generated by most non-private models (11.1%-15% and 18.1%-19.5% accuracy drops compared with GGAN on DBLP and IMDB, respectively). This means even if the attackers identify nodes in the generated (released) networks of DPGGAN, they cannot leverage the information there to accurately infer the existence or absence of links between particular pairs of nodes on the original networks. This directly corroborates our claim that DPGGAN is effective in protecting individual link privacy. Due to space limit, more details and discussions regarding the experimental results are put into Appendix B.

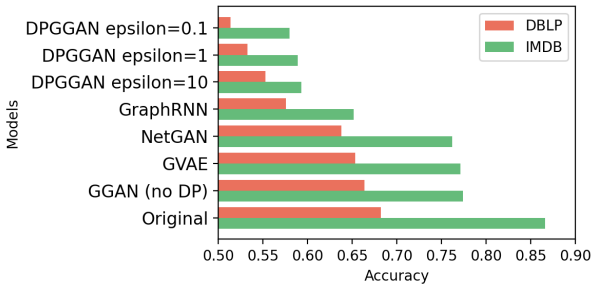


Figure 3: Accuracy of links predicted on networks generated by DPGGAN with varying ε values compared with baselines. *Lower accuracy* means *better individual link privacy*.

5 Conclusion

Due to the recent development of deep graph generation models, synthetic networks are generated and released for granted, without the concern about possible privacy leakage over the original networks used for model training. In this work, for the first time, we provide a compelling system for secure graph generation through the appropriate integration of deep graph generation models and differential privacy. Comprehensive experiments show our model to be effective in both preserving global graph structure and protecting individual link privacy.

References

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

[Barabási and Albert, 1999] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 1999.

[Blocki *et al.*, 2012] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. *FOCS*, 2012.

[Bojchevski *et al.*, 2018] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In *ICML*, 2018.

[Cai *et al.*, 2018] Z. Cai, Z. He, X. Guan, and Y. Li. Collective data-sanitization for preventing sensitive information inference attacks in social networks. *TDSC*, 2018.

[Dwork *et al.*, 2014] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Theoretical Computer Science*, 9(3–4):211–407, 2014.

[Fredrikson *et al.*, 2015] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *SIGSAC*, 2015.

[Frigerio *et al.*, 2019] Lorenzo Frigerio, Anderson Santana de Oliveira, Laurent Gomez, and Patrick Duverger. Differentially private generative adversarial networks for time series, continuous, and discrete open data. In *IFIP SEC*, 2019.

[Gu *et al.*, 2019] Xiaodong Gu, Kyunghyun Cho, Jungwoo Ha, and Sunghun Kim. Dialogwae: Multimodal response generation with conditional wasserstein auto-encoder. In *ICLR*, 2019.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[Kasiviswanathan *et al.*, 2013] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. Analyzing graphs with node differential privacy. In *TCC*, 2013.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Variational graph auto-encoders. In *NIPS Workshop on Bayesian Deep Learning*, 2016.

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

[Kuhn, 2009] Kristine M Kuhn. Compensation as a signal of organizational culture: the effects of advertising individual or collective incentives. *IJHRM*, 20(7):1634–1648, 2009.

[Kurakin *et al.*, 2017] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *ICLR*, 2017.

[Maron *et al.*, 2019] Haggai Maron, Heli Ben-Hamu, Nadav Sharmir, and Lipman Yaron. Invariant and equivariant graph networks. In *ICLR*, 2019.

[Narayanan and Shmatikov, 2009] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. *SP*, 2009.

[Nobari *et al.*, 2014] Sadegh Nobari, Panagiotis Karras, Hwee Hwa PANG, and Stéphane Bressan. L-opacity: Linkage-aware graph anonymization. In *EBDT*, 2014.

[Papernot *et al.*, 2018] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. In *ICLR*, 2018.

[Sala *et al.*, 2011] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y. Zhao. Sharing graphs using differentially private graph models. In *SIGCOMM*, 2011.

[Shokri and Shmatikov, 2015] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *SIGSAC*, 2015.

[Shokri *et al.*, 2017] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *ISP*, 2017.

[Sigurbjörnsson and Van Zwol, 2008] Börkur Sigurbjörnsson and Roelof Van Zwol. Flickr tag recommendation based on collective knowledge. In *WWW*, 2008.

[Simonovsky and Komodakis, 2018] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *ICANN*, 2018.

[Sun and Lyu, 2020] Lichao Sun and Lingjuan Lyu. Federated model distillation with noise-free differential privacy. *arXiv preprint arXiv:2009.05537*, 2020.

[Sun *et al.*, 2018] Lichao Sun, Yingtong Dou, Carl Yang, Ji Wang, Philip S Yu, Lifang He, and Bo Li. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*, 2018.

[Sun *et al.*, 2020a] Lichao Sun, Jianwei Qian, Xun Chen, and Philip S Yu. Ldp-fl: Practical private aggregation in federated learning with local differential privacy. *arXiv preprint arXiv:2007.15789*, 2020.

[Sun *et al.*, 2020b] Lichao Sun, Yingbo Zhou, Philip S Yu, and Caiming Xiong. Differentially private deep learning with smooth sensitivity. *arXiv preprint arXiv:2003.00505*, 2020.

[Wang and Wu, 2013] Yue Wang and Xintao Wu. Preserving differential privacy in degree-correlation based graph generation. *TDP*, 2013.

[Watts and Strogatz, 1998] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *nature*, 393(6684):440, 1998.

[Xie *et al.*, 2020] Yiqing Xie, Sha Li, Carl Yang, Raymond Chi-Wing Wong, and Jiawei Han. When do gnns work: Understanding and improving neighborhood aggregation. In *IJCAI*, 2020.

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

[Xue *et al.*, 2012] Mingqiang Xue, Panagiotis Karras, Raissi Chedy, Panos Kalnis, and Hung Keng Pung. Delineating social network data anonymization via random edge perturbation. In *CIKM*, 2012.

[Yang *et al.*, 2019] Carl Yang, Peiye Zhuang, Wenhan Shi, Alan Luu, and Pan Li. Conditional structure generation through graph variational generative adversarial nets. In *NIPS*, 2019.

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

[You *et al.*, 2018] Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *ICML*, 2018.

[Zhang *et al.*, 2014] Aston Zhang, Xing Xie, Kevin Chen-Chuan Chang, Carl A Gunter, Jiawei Han, and XiaoFeng Wang. Privacy risk in anonymized heterogeneous information networks. In *EDBT*, 2014.

[Zhang *et al.*, 2019] Yanjun Zhang, Xin Zhao, Xue Li, Mingyang Zhong, Caitlin Curtis, and Chen Chen. Enabling privacy-preserving sharing of genomic data for gwass in decentralized networks. In *WSDM*, 2019.