Network Generation with Differential Privacy

Xu Zheng
Accenture Labs
Dublin, Ireland
xu.b.zheng@accenture.com

Nicholas McCarthy
Accenture Labs
Dublin, Ireland
nicholas.mccarthy@accenture.com

Jer Hayes
Accenture Labs
Dublin, Ireland
jeremiah.hayes@accenture.com

Abstract

We consider the problem of generating private synthetic versions of real-world graphs containing private information while maintaining the utility of generated graphs. Differential privacy is a gold standard for data privacy, and the introduction of the differentially private stochastic gradient descent (DP-SGD) algorithm has facilitated the training of private neural models in a number of domains. Recent advances in graph generation via deep generative networks have produced several high performing models. We evaluate and compare state-of-the-art models including adjacency matrix based models and edge based models, and show a practical implementation that favours the edge-list approach utilizing the Gaussian noise mechanism, when evaluated on commonly used graph datasets. Based on our findings, we propose a generative model that can reproduce the properties of real-world networks while maintaining edge-differential privacy. The proposed model is based on a stochastic neural network that generates discrete edge-list samples and is trained using the Wasserstein GAN objective with the DP-SGD optimizer. Being the first approach to combine these beneficial properties, our model contributes to further research on graph data privacy.

1 Introduction

Advances in generative models for graphs driven by the rise of deep learning have led to a number of proposed applications including data imputation, novel molecule generation, and knowledge discovery [1, 2]. Several works have used generative adversarial network (GAN) architectures [3] in the problem of generating synthetic versions of real-world graphs by operating directly on adjacency matrices [4, 5]. However, this approach has a number of problems: foremost it has limited scalability as the computation and memory requirements scale quadratically with the number of graph vertices, making it feasible to process only small graphs. Moreover, they have mixed results when attempting to generate graphs that match the statistical properties of real-world graphs [6]. Node permutation can be used to create different variants of the graph in order to train a GAN, but this can significantly affect graph structure resulting in difficulties training the generator, requires node ordering heuristics or other graph matching operations.

Node embedding methods achieve state-of-the-art scores in tasks like link prediction and node classification [1, 7, 8]. The main idea behind these approaches is to model the probabilities of each edge's existence by a neural network. Nevertheless, such approaches still have difficulty in preserving and generating patterns inherent to real-world networks. The NetGAN model [9] applies random walks on the input graphs [10], and an LSTM-based generator tries to mimic these walk

Preprint. Work in progress.

sequences [11]. As it considers only the non-zero entries of the adjacency matrix it avoids the heavy computational requirements for operating on adjacency matrices, and is readily applicable to graphs with thousands of nodes.

Although edge-based GAN models can synthesize graphs that closely model the original data distribution, the existence of private data in the graph is not taken into account, and the generated graph can breach individual privacy by leaking personally identifiable information. Among research communities and in certain domains such as medical, healthcare and insurance there is a significant advantage to generating synthetic representations of private data. However, synthetic data generated using standard generative model architectures is not private [12]. Privacy-preserving release of tabular and image data [13] has been shown to be possible by implementing a generative adversarial framework with differential privacy mechanisms [14, 15]. Hence, integrating generative models and differential privacy mechanisms for generating graphs that can maintain both statistical and topological information of the original graphs opens new avenues for further research. The contributions of our work are as follows. Firstly we propose a new model for generating graphs using an edge list. This model is able to learn from a single graph and generate discrete outputs that maintain topological properties of the original graph. Secondly, by engaging the edge list we successfully integrated differential privacy mechanisms into the training procedure. Theoretically, we show that our algorithm ensures differential privacy for the generated graphs, and demonstrate empirically its performance in experiments on two datasets. The results show that our method is competitive with baseline model NetGAN [9] in terms of both statistical properties and data utility.

2 Method

In this section we present our model for synthesizing graphs with edge-differential privacy. The overall architecture is based on NetGAN [9], which operates random walks on the input graph [10]. We modify the training procedure by replacing the random walk method and instead represent the input graph G by a sparse adjacency tuple I, where $I \in \mathbb{N}^2$ encodes edge indices in COOrdinate format [16]. By modifying and using an edge list instead of random walk, we can train the generative model with edge-differential privacy. As with typical GAN architectures, this model contains two main components: a generator g, which is trained to generate synthetic edge lists, and a discriminator d that learns to distinguish the synthetic edge lists from the real ones that are sampled from the real set of edges. We train our model based on the Wasserstein GAN (WGAN) framework because it is more stable and able to prevent mode collapse [17]. The architecture can found in Figure 1, and the gradient perturbation algorithm can be found in algorithm 1.

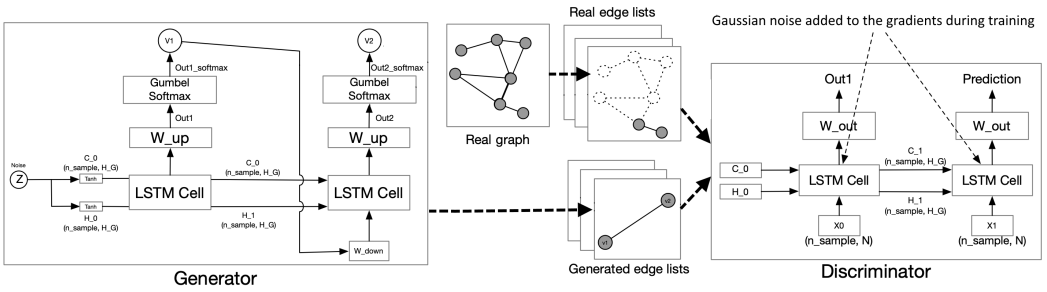

Figure 1: The architecture of DP-NetGAN. The Generator and Discriminator are based on an LSTM architecture, and the inputs to the model are the edge lists of the graph. Calibrated noise is added to the discriminator during training.

2.1 Differential Privacy

Differential privacy can be achieved by injecting Gaussian noise parameterized by sensitivity to the clipped gradients in the optimization procedure when training d. In this work we use the differentially private stochastic gradient descent (DP-SGD) algorithm 1 to sanitize the gradients of discriminator d and add calibrated Gaussian noise. The noise is scaled by the noise factor σ defined in algorithm 1 and the clip value C applied on the gradients represents the sensitivity in differential privacy mechanisms.

A key component of the model is to keep track of the cumulative privacy loss during the training phase. Due to the composability property of differential privacy, each training step's privacy cost can be accumulated by a privacy accountant. We employ the Moments Accounting procedure proposed in [18] that additively accumulates the log of the moments of the privacy loss at each step. This method provides a tighter estimation of the privacy loss and allows us to compute the overall privacy costs at each iteration.

2.2 Assembling Graph from Edge List

Following these steps we can use an edge list to compose an adjacency matrix that represents a synthetic graph, illustrated in Figure 2. We firstly use the trained generator g to generate a large number of edge lists. In work [9] it has been shown that a larger number here results in better graphs. A count of how often an edge appears in the set of generated edge lists is used to create a square matrix S. In this matrix we can not guarantee the symmetry property in the input graph as it is undirected, so we set $s_{ij} = s_{ji} = max(s_{ij}, s_{ji})$ explicitly. If we just apply a threshold on the matrix to construct an adjacency matrix, nodes with a high degree tend to be over-represented and low degree nodes can be left out. Hence, we firstly ensure every node i has at least one edge by sampling a neighbor j with probability $p_{ij} = \frac{s_{ij}}{\sum_v Siv}$. We continue sampling for each edge (i, j) with the probability $p_{i,j} = \frac{s_{ij}}{\sum_{u,v} Suv}$ until we reach as many unique edges as in the original graph. For each (i, j) we also include (j, i) so that we can have an undirected graph. We can then convert this raw counts matrix to a binary adjacency matrix.

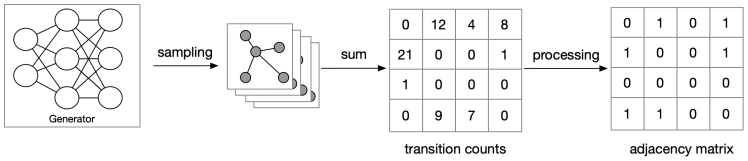

Figure 2: Constructing an adjacency matrix from raw edge lists output by the generator.

3 Experiments and Results

We evaluate the quality of the generated graphs from two aspects: graph statistical properties and graph utility. The graph statistical properties include the graph maximum degrees, assortativity, triangle count, the power-law exponent, clustering coefficient and characteristic path length (definitions in Appendix 3). Graph utility is evaluated based on link prediction performance on the validation set. We chose two well-known datasets for our experiments, CORA-ML [19] and Citeseer [20]. For these experiments we treat the graphs as undirected and only consider the largest connected component (LCC) so that we can compare the results with the baseline NetGAN model [9], which requires the graph to be connected. We randomly sample 15% of the edges as a validation set, and fit two models to the remaining training set. We trained NetGAN with the standard 16-step random walk and a 2-step random walk for comparison with the edge list approach. Early stopping is used to stop the model training if performance on the validation set does not increase in previous 5 checkpoints As the evaluation process is time consuming and resource consuming, each checkpoint contains m epochs, while m is a parameter set before training.

3.1 Graph Statistics

In our experiments we set the *batch size* = 2048 and δ = 10e-5 for both datasets. Gradient clip values are selected from {0.05, 0.1, 0.5, 1}, as we observed that most of the gradients of the discriminator are smaller than 1 for the standard NetGAN model. The value of ϵ is then computed for different noise levels and training epochs E with the moments accountant. For both DP-NetGAN and NetGAN, we generate 400k edge lists or walks and construct the adjacency matrix as the final generated graph at each checkpoint. Graph statistics are calculated on these generated graphs with different level of epsilon values. In Table 1, we observe that DP-NetGAN captures the graph properties well, and note that the increasing epsilon values result in a relatively stable improvement in graph statistics: the privacy-utility trade-off. It's worth noting that NetGAN with 16-steps achieves highest performance

over all metrics, but that DP-NetGAN with $\epsilon = \inf$ surpasses NetGAN with 2 steps due to the change in training protocol. This is most obvious the #Triangles statistic as the 2-step random walk fails to capture local triangular patterns of the input graph.

To increase edge privacy, we can increase the noise added during training, resulting in smaller epsilon values. Although the edge overlap between input and generated graphs is relatively small (<20%) for DP-NetGAN, the graph statistics are comparable, implying that DP-NetGAN does not just memorize a subset of edges and randomly generate the remainder, but instead captures the underlying structure of the network.

Table 1: Table of graph statistics on the output of models at different epsilon values.

Model / Data	**MaxDeg.**	**Assor.**	**#Triangle**	**PLE**	**Cluster**	**CPL**
CORA-ML	**240**	**-0.075**	**2814**	**1.86**	**2.7e-3**	**5.61**
DP-NetGAN(ϵ=2.9)	200	-0.258	386	1.84	2e-4	3.81
DP-NetGAN(ϵ=9.5)	138	-0.134	393	1.86	7.6e-4	4.00
DP-NetGAN(ϵ=440)	249	-0.036	829	1.72	6.7e-4	4.32
DP-NetGAN(ϵ=inf)	220	-0.058	807	1.72	1e-3	4.33
NetGAN(2steps)	151	-0.101	249	1.68	7.5e-4	4.40
NetGAN(16steps)	262	-0.065	1830	1.71	1.4e-3	4.48
CITESEER	**99**	**0.0075**	**1084**	**2.07**	**1.3e-2**	**9.33**
DP-NetGAN(ϵ=0.68)	400	-0.212	434	2.28	7e-4	4.9
DP-NetGAN(ϵ=16)	276	-0.075	78	2.18	6.35e-5	4.69
DP-NetGAN(ϵ=296)	74	-0.052	26	2.03	6.34e-4	5.5
DP-NetGAN(ϵ=inf)	54	-0.024	155	1.96	7.3e-3	6.28
NetGAN(2steps)	32	-0.167	54	1.96	5.3e-3	6.19
NetGAN(16steps)	55	-0.025	493	2.02	1.7e-2	6.88

3.2 Link Prediction.

Link prediction is a standard task in graph learning where the goal is to estimate the probability of links between nodes in a graph. We use the area under the ROC curve (AUC) and average precision (AP) to evaluate the link prediction performance on the validation set. We can see from table 2 that our model shows competitive performance for two datasets compared with NetGAN 2-steps, and our model's performance increases when we increase the ϵ values.

Table 2: Link prediction performance on validation set.

	Cora-ML		Citeseer	
Model	AUC	AP	AUC	AP
DP-NetGAN (ϵ=*1e-1)	60.12	60.07	69.68	68.73
DP-NetGAN (ϵ=*1e2)	80.04	80.49	72.03	75.42
DP-NetGAN (ϵ=*1e3)	89.29	90.73	89.29	90.73
DP-NetGAN (ϵ=inf)	88.44	89.9	83.47	86.18
NetGAN (2steps)	85.33	86.72	80.88	83.76
NetGAN (16steps)	96.17	96.65	95.33	96.38

4 Discussion and Future Work

In this work we introduced a scalable differentially private generative model for graph data. By utilizing edge lists, this model can generate networks that capture essential topological properties. It also shows competitive link prediction performance while maintaining edge-differential privacy. Future research directions could look at private synthesis of multi-relational graph data such as knowledge graphs, which could be achieved by a relatively simple modification of the architecture proposed here by introducing a relation step into the LSTM input, and embedding layers for node and edge to maintain corresponding input dimensions. Alternatively, examining differential privacy mechanisms such as PATE in a graph generation context would also be of use.

References

[1] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[2] Nicola De Cao and Thomas Kipf. MolGAN : An implicit generative model for small molecular graphs. 2018.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[4] Sahar Tavakoli, Alireza Hajibagheri, and Gita Sukthankar. Learning social graph topologies using generative adversarial neural networks. In *International Conference on Social Computing, Behavioral-Cultural Modeling & Prediction*, 2017.

[5] Shuangfei Fan and Bert Huang. Labeled graph generative adversarial networks. *CoRR*, abs/1906.03220, 2019.

[6] Yuxiao Dong, Reid A. Johnson, Jian Xu, and Nitesh V. Chawla. Structural diversity and homophily: A study across more than one hundred big networks. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Part F129685:807–816, 2017.

[7] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[8] Aditya Grover and Jure Leskovec. node2vec : Scalable Feature Learning for Networks. pages 855–864, 2016.

[9] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 609–618, 2018.

[10] László Lovász et al. Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty*, 2(1):1–46, 1993.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. 2018.

[13] Sean Augenstein, H Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, et al. Generative models for effective ml on private, decentralized datasets. *ICLR*, 2020.

[14] Cynthia Dwork and Vitaly Feldman. Privacy-preserving prediction. *Conference on Learning Theory (COLT)*, 2018.

[15] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018.

[16] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 214–223, 2017.

[18] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, 2016.

[19] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[20] 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.

A Graph Statistics

Table 3: Graph Statistics List

Metric name	Description
Max Degree	Maximum degree of all nodes in a graph
Assortativity	Pearson correlation of degrees of connected nodes
Triangle Count	Number of triangles in the graph
Power law exponent	Exponent of the power law distribution
Clustering coeff	The degree to which nodes in a graph tend to cluster together
Characteristic path len	Average number of steps along the shortest paths for all possible pairs of nodes

B DP GAN

Algorithm 1: Basic DP GAN

input : n - number of samples; λ - coefficient of gradient penalty; n_{critic} - number of critic iterations per generator iteration; n_{param} - number of discriminator's parameters; m - batch size; $(\alpha, \beta_1, \beta_2)$ - Adam hyper-parameters; C - gradient clipping bound; σ - noise scale;

output : A differentially private generator G

while θ *has not converged* **do**

  for $i \leftarrow 1$ **to** n_{critic} **do**

    for $j \leftarrow 1$ **to** m **do**

      sample $x \sim p_{data}, z \sim p_z, \rho \sim \mu[0,1]$;

      $\hat{x} \leftarrow px + (1-p)G(z)$;

      $l^{(i)} \leftarrow D(G(z)) - D(x) + \lambda(\|\Delta_{\hat{x}} D(\hat{x})\|_2 - 1)^2$;

      $g^{(i)} \leftarrow \Delta_w l^{(i)}$;

      $g^{(i)} \leftarrow g^{(i)} / max(1, \|g^{(i)}\|_2 / C)$;

    // Gaussian noise ;

    $\xi \sim N(0, (\sigma C)^2 I)$;

    $\hat{g} = \frac{1}{m}(\sum_{i=1}^{m} g^{(i)} + \xi)$;

    $w \leftarrow Adam(\hat{g}, w, \alpha, \beta_1, \beta_2)$;

    update A with (θ, m, n_{param}) ;

  sample $\{z^{(i)}\}_{i=1}^{m} \sim p_z$;

  // updating generator ;

  $\theta \leftarrow Adam(\Delta_\theta \frac{1}{m} \sum_{i=1}^{m} -D(G(z^{(i)})), \theta, \alpha, \beta_1, \beta_2)$;

  // computing cumulative privacy loss ;

  $\delta \leftarrow$ query A with ϵ_0 ;

  if $\delta > \delta_0$ **then**

    break

return G