BRGAN: Generating Graphs of Bounded Rank

William Shiao
University of California Riverside
wshia002@ucr.edu

Evangelos E. Papalexakis
University of California Riverside
epapalex@cs.ucr.edu

ABSTRACT

Graph generation is a task that has been explored with a wide variety of methods. Recently, several papers have applied Generative Adversarial Networks (GANs) to this task, but most of these methods result in graphs of full or unknown rank. Many real-world graphs have low rank, which roughly translates to the number of communities in that graph. Furthermore, it has been shown that taking the low rank approximation of a graph can defend against adversarial attacks. This suggests that testing models against graphs of different rank may be useful. However, current methods provide no way to control the rank of generated graphs. In this paper, we propose BRGAN: a GAN architecture that generates synthetic graphs, which in addition to having realistic graph features, also have bounded (low) rank. We extensively evaluate BRGAN and show that it is able to generate synthetic graphs competitive with state-of-the-art models, with rank equal to or lower than the desired rank.

KEYWORDS

graph generation, generative adversarial networks, graph rank

ACM Reference Format:
William Shiao and Evangelos E. Papalexakis. 2020. BRGAN: Generating Graphs of Bounded Rank. In *Proceedings of Machine Learning on Graphs (MLG'2020).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Graph generation is a common task, and many models exist for this task. Traditional statistical methods (such as the Barabási–Albert model [1]) usually attempt to model specific attributes of the graph. While this can lead to generated graphs being similar to the target graph with respect to one or more attributes, it requires the user to accurately identify characteristics of the graph they wish to mimic. This issue has been addressed with the introduction of neural network models (like GraphRNN [26]), which learn directly from a set of graphs without requiring the user to explicitly declare any graph attributes.

Recent advances in the field include the use of Generative Adversarial Networks (GANs) for graph generation. The GAN was first introduced in 2014 by Goodfellow *et al.*[11]. While they were originally used for image generation, they have since been applied to a wide variety of fields, including the field of graph generation.

A current state-of-the-art model LGGAN [7] learns to directly generate the adjacency matrices of graphs and its corresponding labels given a set of graphs as input. It does this by using a GAN with a Graph Convolutional Network (GCN) [18] as its discriminator and a Multi-layer Perceptron (MLP) network as its generator.

MLG'2020, August 2020, San Diego, CA
2020. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

There are also several other deep learning methods. GraphRNN [26] models the graph as a sequence and uses a RNN to generate realistic graphs. GraphVAE [21] uses a Variational Autoenconder (VAE) to generate graphs.

However, most of these existing graph generation techniques tend to produce a graph of full or high rank. While this is sufficient for most use cases, generating a graph of a known rank can also be useful. The rank of a community graph roughly corresponds to the number of communities within the graph. Bounding the rank of the generated graph therefore allows a user to generate graphs with bounded number of communities, which is difficult or impossible with existing graph generation models.

Low-rank approximations of graphs have also been shown to be useful in defending against adversarial attacks [5], which further suggests that the rank of a graph is a useful parameter. Being able to control the rank allows users to generate synthetic datasets with realistic graphs to test the effectiveness of new models on inputs with different ranks.

Determining the full rank of a matrix is computationally tractable, done by the Singular Value Decomposition (SVD). However, the problem becomes difficult when we move to higher-order structures like time evolving graphs, since finding the rank of three-dimensional tensor is NP-hard [14]. Generating realistic data where the rank of the data is known can further promote research in developing and testing methods like AutoTen [19] or NSVD [22] for approximating the rank of a tensor. We reserve this for future work, however, it serves as compelling motivation for the generation of realistic matrices and higher-order tensors of known rank.

This work focuses on generating graphs (adjacency matrices) of fixed rank, rather than tensors, because the SVD can be used to find the approximate rank of a matrix in polynomial time. This allows us to evaluate the effectiveness of our bound on the rank of the graph. We can also evaluate the quality of our generated graphs by comparing it against other established models.

We propose a new model, BRGAN, that first generates *factor* matrices of rank of at most ρ and uses these factor matrices to construct the adjacency matrix. This allows us to bound the rank of the generated graphs.

Our contributions include:

- **Novel problem formulation**: we propose the problem of adversarially generating an adjacency matrix with a desired rank.
- **Novel architecture**: we propose a novel architecture for the above problem that generates the adjacency matrix from factor matrices.
- **Experimentation**: we evaluate the effectiveness of our approach along two dimensions: the rank of the generated graphs and the realism of the generated.

2 PROBLEM FORMULATION & PROPOSED METHOD

In this work, we generate a graph of bounded rank. We define the rank of a graph as the rank of its adjacency matrix. Recall that the rank of a matrix is equal to the number of rank-1 matrices required to sum up to it.

SVD can be used to find a rank-k approximation of an adjacency matrix which, by the Eckart–Young–Mirsky theorem, is optimal with respect to the Frobenius norm of the difference between the rank-k approximation and original matrix. However, this approximation will not necessarily preserve graph properties. Our goal is to generate a realistic graph from input graphs that will exhibit similar graph properties to those of the inputs.

Current graph generation models produce graphs of unbounded rank. This is because existing methods tend to either model the graph as a sequence or generate the adjacency matrix directly, which makes it difficult to bound the rank of the resulting matrix.

Let the two factor matrices be denoted $\mathbf{A}$ and $\mathbf{B}$, where $\mathbf{A} \in \mathbb{R}^{n \times \rho}$ and $\mathbf{B} \in \mathbb{R}^{\rho \times n}$. Then, their product $\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{n \times n}$ and $\text{rank}(\mathbf{C}) \leq \min(\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B}))$. By definition, $\text{rank}(\mathbf{A}) \leq \min(n, \rho)$ and $\text{rank}(\mathbf{B}) \leq \min(n, \rho)$. If we define ρ such that $\rho \leq n$, we know that $\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B}) \leq \rho$. Then, $\text{rank}(\mathbf{C}) \leq \rho$. We exploit this property to generate a graph of bounded rank. Note that this approach does not guarantee that $\mathbf{C}$ is of exactly rank ρ because the vectors in $\mathbf{A}$ and $\mathbf{B}$ are not guaranteed to be linearly independent.

Then, we simply have to generate the two factor matrices $\mathbf{A}$, $\mathbf{B}$ and use those two to generate the output matrix C. We can then calculate the loss and use that to backpropagate as usual.

2.1 Architecture

A GAN consists of two main models: the generator $\mathcal{G}$ and the discriminator $\mathcal{D}$. The generator maps a sample from a space (z) to an adjacency matrix $\mathbf{M}$. The discriminator takes an adjacency matrix $\mathbf{M}$ and outputs the probability p of a sample being fake. These two models are then trained in unison in the hope that the generator will improve at generating realistic graphs and the discriminator will improve in identifying fake graphs. We use the CT-GAN [25] framework, which improves upon WGAN-GP [13], which is in turn an improvement on WGAN [2].

$\mathcal{G}$ consists of a multi-layer perceptron (MLP) network followed by two MLP networks. The two MLP networks take the output of the first MLP network and use it to generate the factor matrices. This reduces the number of parameters and results in faster convergence and better performance when compared to a single MLP network.

$\mathcal{D}$ accepts an $n \times n$ adjacency matrix and consists of several Graph Convolutional Network (GCN) layers followed by a single fully-connected layer. We take advantage of the residual GCN connections by performing a max pool across all of the GCN layer outputs. As Fan and Huang [7] found, the network performs better with these residual connections.

We apply a scaled hard tanh to the output of each network to clip the output values to $[0, 1]$. While this does not cause each entry in the final adjacency matrix M to be bounded to $[0, 1]$, it does help reduce the chance of the discriminator easily detecting fakes based on the value of the nodes, rather than on the structure of the graph. This helps ensure that the model continues training and the discriminator loss does not reach 0. We tested both the scaled hard tanh and a sigmoid and found no major differences.

3 EXPERIMENTAL EVALUATION

In evaluating our model, we try to answer two different questions. First, can our model generate graphs of a specific rank? Second, do those graphs mimic real-world graphs?

3.1 Methodology

We evaluated our model using egonets extracted from the CORA and Citeseer citation graph datasets [20]. We split the graph into a small dataset, which consists of 2-egonets and 3-egonets with 30-50 nodes. This was also used to evaluate LGGAN [7], which allows us to benchmark our results against theirs.

Note that while these are the same base datasets as in You *et al.* [26], we sample the egonets differently since we wish to compare our results with Fan and Huang [7] (who also use the same datasets).

While BRGAN's discriminator uses a GCN and is node permutation invariant, BRGAN's generator is affected by node ordering because it uses a MLP. As such, we use the approach used by You *et al.*[26] and Fan *et al.*[7] and generate all possible BFS orderings of the graph. This allows use to only have n^2 permutations per graph rather than the full $n!$ possible permutations. We then train on this augmented dataset.

Both the generator and discriminator were trained with the RMSprop [15] optimizer because we found that it tends to be more stable than ADAM [17] for this case. Arjovsky *et al.*[2] also suggested using RMSprop for WGANs, although Wei *et al.*[25] uses ADAM. We did not perform a hyperparameter search for the learning rate and found that our model generally works well with any reasonable learning rate. For the CTGAN loss function, we used $\lambda_1 = 10$ and $\lambda_2 = 2$: the same values as Wei *et al.*[25].

The output of $\mathcal{G}$ is an adjacency matrix M with values in $[0, r]$, although values tend to be in the range $[0, 1]$ due to the input values also being in this range. The graphs are then thresholded by some threshold τ. That is:

$$\mathbf{M}_{i,j} = \begin{cases} 0, & \text{if } \mathbf{M}_{i,j} \leq \tau \\ 1, & \text{otherwise} \end{cases} \tag{1}$$

τ controls the sparsity of the final graph. As $\tau \longrightarrow 0$, $|V| \longrightarrow 0$ and as $\tau \longrightarrow r$, $|V| \longrightarrow m \cdot n$. For our tests, we chose $\tau = 0.5$ because the majority of the values are in the range $[0, 1]$ and $\tau = 0.5$ provides a mid-way bound.

We found that some of the graphs generated by $\mathcal{G}$ sometimes had disconnected vertices. Disconnected vertices are a problem because the clustering coefficient of a disconnected graph is infinity.

The disconnected vertices were likely caused by the fact that we used different graph sizes for our input dataset. This means that we had to zero-pad the adjacency matrix of the dataset's graphs to feed it into the network. We chose to zero-pad rather than to pad it with the identity matrix because this results in more sparsity in the input graph and encourages a sparser output representation. To solve the issue with disconnected vertices, we remove any vertices

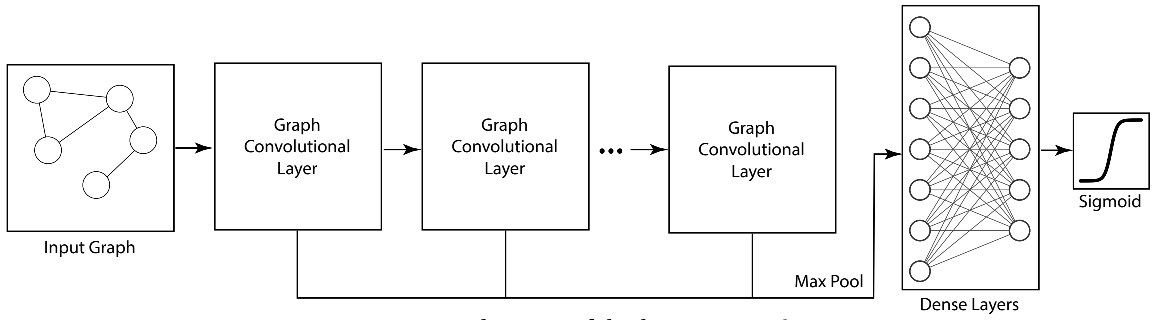
Input Graph
Graph Convolutional Layer
Graph Convolutional Layer
Graph Convolutional Layer
Max Pool
Dense Layers
Sigmoid

Figure 1: Architecture of the discriminator $\mathcal{D}$.

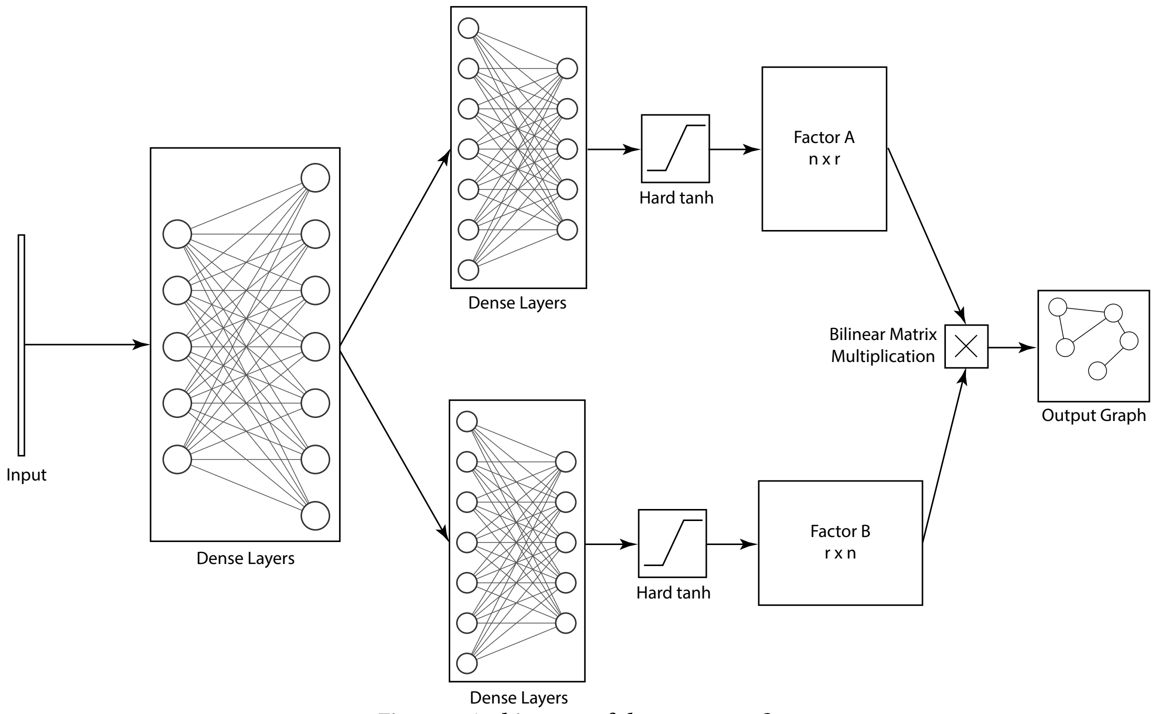

Figure 2: Architecture of the generator $\mathcal{G}$.

that had no edges on both the input and generated graphs before calculating any statistics.

Singular Value Decomposition (SVD) can be used to find the approximate rank of a matrix. The SVD of a matrix **M** is as follows:

$$\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^T \tag{2}$$

where **M** is a $m \times m$ orthogonal matrix, Σ is a diagonal non-negative $m \times n$ matrix, and **V** is a $n \times n$ orthogonal matrix. By convention, the singular values $\sigma_1, \sigma_2, \ldots \sigma_m$ are sorted in descending order by value. Then, the approximate rank of a matrix can be found by finding the smallest index k where $\sigma_{k+1} < \epsilon$, where ϵ is some small constant. In our experiments, we found that $\epsilon = 10^{-6}$ is sufficient.

3.2 Experiental Results

We evaluated the realism of the generated graphs. To do so, we compared several graph statistics of the input graphs with those of the output graphs. We did this by calculating the Mean Maximum Discrepancy (MMD) [12] between the degree distribution, clustering coefficient, and orbits of the two sets of graphs. We compared this to the results of various other graph generation methods in the tables below. The source code for LGGAN was not available at the time of writing, so we used the results reported in their paper [7].

While the results vary with the value of ρ, we can see that BRGAN generally has similar degree and orbit MMDs when compared to LGGAN. However, it tends to have a higher clustering MMD. This is likely because lowering rank has the largest effect (out of the three tested graph statistics) on the clustering coefficient of a graph. This is because of the fact that we are imposing low-rank structure on the adjacency matrix of the graph. Then, all of the nodes live in a much lower dimension, which increases the density of connections and therefore would also increase the clustering coefficient.

We also evaluated the bound on the rank of the generated graphs and, as theoretically expected, we found that our bound was effective and the rank of generated matrices were equal to or less than ρ. We also found that as ρ increases, the median rank of the generated graphs generally also increases. However, the median rank of the generated matrices is closer to ρ when ρ is small. This means that ρ becomes a tighter bound on the true rank of the generated graphs as $\rho \rightarrow 1$.

This is because it is difficult to create a realistic approximation of a high-rank matrix while maintaining low rank. However, as the rank bound is increased, the median generated rank does not increase linearly. This is because some of the input graphs are already of low rank and therefore some of the generated graphs will not be of higher rank, leading to a lower median generated graph rank.

We also noticed that the MMD of our graph attributes do not always uniformly improve as ρ increases. This is likely because that, as the number of parameters increases as ρ increases. However, the clustering MMD does improve in almost all cases as ρ increases.

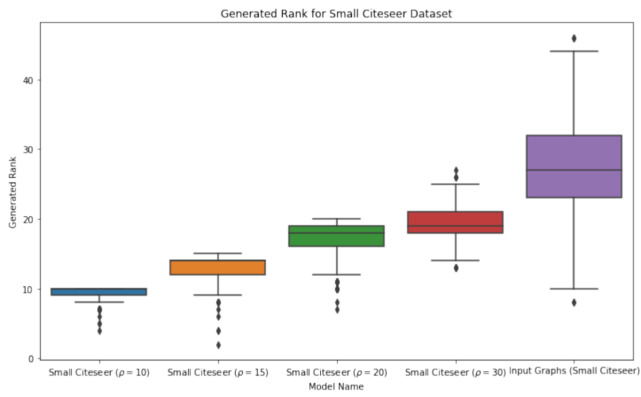

Figure 3: Barplot of the true rank of generated graphs from the small Citeseer dataset with $\rho = 10, 15, 20, 30$. The rightmost bar represents the distribution of the original dataset. We can observe that the bound on the rank is effective.

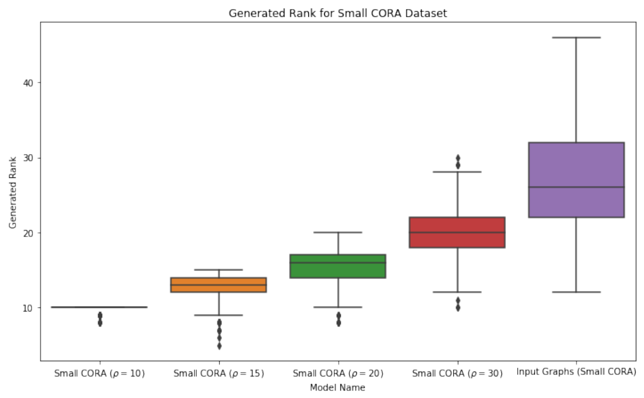

Figure 4: Distribution of the true rank of generated graphs from the small CORA dataset with $\rho = 10, 15, 20, 30$. The rightmost bar represents the distribution of the original dataset. We can observe that the bound on the rank is effective.

Model Name	Deg.	Clust.	Orbit
Erdos-Renyi	0.68	0.94	0.48
Barabási–Albert	0.31	0.53	0.11
MMSB	0.21	0.68	0.07
DeepGMG	0.34	0.44	0.27
GraphRNN	0.26	0.38	0.39
LGGAN	0.13	0.08	0.03
BRGAN (ρ = 10)	0.02	0.64	0.07
BRGAN (ρ = 15)	0.02	0.56	0.04
BRGAN (ρ = 20)	0.03	0.47	0.07
BRGAN (ρ = 30)	0.09	0.39	0.11
BRGAN (ρ = 50)	0.07	0.34	0.11

Table 1: Evaluation results for the small CORA dataset. The degree and orbit MMD values are largely competitive with the other generative models, but LGGAN has a significantly lower clustering MMD value. Note that the numbers for other models are the ones reported by Fan and Huang [7]

Model Name	Deg.	Clust.	Orbit
Erdos-Renyi	0.63	0.86	0.12
Barabási–Albert	0.37	0.18	0.11
MMSB	0.17	0.5	0.11
DeepGMG	0.27	0.36	0.2
GraphRNN	0.19	0.2	0.39
LGGAN	0.17	0.13	0.04
BRGAN (ρ = 10)	0.08	0.18	0.12
BRGAN (ρ = 15)	0.11	0.22	0.08
BRGAN (ρ = 20)	0.07	0.23	0.08
BRGAN (ρ = 30)	0.09	0.28	0.07
BRGAN (ρ = 50)	0.03	0.27	0.05

Table 2: Evaluation results for the small Citeseer dataset. The degree MMD values are much better than that of other models, but the clustering coefficient MMD are generally higher than that of LGGAN. Note that the numbers for other models are the ones reported by Fan and Huang [7]

4 RELATED WORK

Graph generation has been widely studied, and many different models exist for this task. Traditional statisical models such as the Barabási–Albert [1] and exponential random graph models are created to model specific graph properties and tend to work well for only specific use cases.

One such example is the Erdös-Rényi-Gilbert random graph model [6] [9]. The model is often referred to in the form $G(N, p)$ or $G(N, E)$, where N is the number of nodes, E is the number of edges, and p is the probability of an edge existing between two nodes. One of the key properties of this model is that expected number of neighbors for each node is the same [10]. However, this property is often not true of real-world graphs.

Several models attempt to address this limitation by attempting to vary node degree and mimic more realistic node distributions. Notably, the Barabási–Albert model attempts to address this issue by using preferential attachment, which mimics the evolution of a scale-free graph over time and results in the creation of "hubs" in the graph.

Another common family of statistical graph generation models are the exponential random graph models (ERGMs). An example of this are Markov graphs, for which Frank and Strauss [8] proved the probability distributions. This was later generalized by Wasserman and Pattison [24] in the form of p^* models. With Markov chain Monte Carlo (MCMC) proposed by Hunter and Handcock [16], it is possible to estimate the parameters of ERGMs and generate similar graphs.

However, these statistical models all share the weakness that they attempt to model specific graph properties. Recently, advances in neural networks and deep learning have led the creation of several new models that learn directly from an input distribution of one or more graph(s). Unlike traditional models, these methods attempt to learn the structure of a graph rather than simply attempting to match specific attributes. Some notable methods include GraphRNN [26], NetGAN [3], MolGAN [4].

NetGAN [3] takes in a graph and learns the distribution of biased random walks using a LSTM, which allows it to easily scale to large graphs. MolGAN [4] uses a reinforcement learning objective with a reward network (in addition to the standard discriminator and generator) to generate realistic molecular graphs. GraphGAN [23] proposes a novel graph softmax function and learns the connectivity distribution over the vertices, but does not directly generate similar graphs.

However, none of these models provide a method to bound the rank of the generated graph. BRGAN allows a user to easily bound the rank of the generated graphs while maintaining competitve performance.

5 CONCLUSION

In this work, we propose the Bounded Rank GAN (BRGAN), which generates graphs or rank equal to or lower than a hyperparameter ρ. We discuss the merits and limitations of this approach. Finally, we thoroughly evaluate the performance of BRGAN and show that it has competitive performance with existing models and that it provides an effective bound on the rank of generated graphs.

REFERENCES

[1] Reka Albert and Albert-Laszlo Barabasi. 2001. Statistical mechanics of complex networks. (6 2001). DOI: http://dx.doi.org/10.1103/RevModPhys.74.47

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. (1 2017). http://arxiv.org/abs/1701.07875

[3] Aleksandar Bojchevski and Oleksandr Shchur. 2018. *NetGAN: Generating Graphs via Random Walks.* Technical Report. https://www.kdd.in.tum.de/netgan

[4] Nicola De Cao and Thomas Kipf. 2018. MolGAN: An implicit generative model for small molecular graphs. (5 2018). https://arxiv.org/abs/1805.11973

[5] Negin Entezari, Saba A. Al-Sayouri, Amirali Darvishzadeh, and Evangelos E. Papalexakis. 2020. All you need is Low (rank): Defending against adversarial attacks on graphs. In *WSDM 2020 - Proceedings of the 13th International Conference on Web Search and Data Mining.* Association for Computing Machinery, Inc, New York, NY, USA, 169–177. DOI: http://dx.doi.org/10.1145/3336191.3371789

[6] P Erdős and A Rényi. 1959. On random graphs. *Publicationes Mathematicae* (1959).

[7] Shuangfei Fan and Bert Huang. 2019. Labeled Graph Generative Adversarial Networks. (6 2019). http://arxiv.org/abs/1906.03220

[8] Ove Frank and David Strauss. 1986. Markov Graphs. *J. Amer. Statist. Assoc.* 81, 395 (9 1986), 832–842. DOI: http://dx.doi.org/10.1080/01621459.1986.10478342

[9] E. N. Gilbert. 1959. Random Graphs. *The Annals of Mathematical Statistics* 30, 4 (12 1959), 1141–1144. DOI: http://dx.doi.org/10.1214/aoms/1177706098

[10] Anna Goldenberg, Alice X Zheng, Stephen E Fienberg, and Edoardo M Airoldi. 2009. A survey of statistical network models. (12 2009). http://arxiv.org/abs/0912.5410

[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. (6 2014). http://arxiv.org/abs/1406.2661

[12] Arthur Gretton, Karsten M. Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alexander J. Smola. 2007. A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems.* DOI: http://dx.doi.org/10.7551/mitpress/7503.003.0069

[13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved Training of Wasserstein GANs. (3 2017). http://arxiv.org/abs/1704.00028

[14] Johan Håstad. 1990. Tensor rank is NP-complete. *Journal of Algorithms* 11, 4 (12 1990), 644–654. DOI: http://dx.doi.org/10.1016/0196-6774(90)90014-6

[15] Geoffrey E. Hinton, Nitish Srivastava, and Kevin Swersky. 2012. Lecture 6a-overview of mini-batch gradient descent. *COURSERA: Neural Networks for Machine Learning* (2012).

[16] David R Hunter and Mark S Handcock. 2006. Inference in Curved Exponential Family Models for Networks. *Journal of Computational and Graphical Statistics* 15, 3 (9 2006), 565–583. DOI: http://dx.doi.org/10.1198/106186006X133069

[17] Diederik P Kingma and Jimmy Lei Ba. *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION.* Technical Report. https://arxiv.org/pdf/1412.6980.pdf

[18] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. (9 2016). http://arxiv.org/abs/1609.02907

[19] Evangelos E Papalexakis. Automatic Unsupervised Tensor Mining with Quality Assessment. (????). http://www.cs.ucr.edu/~epapalex/papers/sdm16-autoten.pdf

[20] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Magazine* (2008). DOI: http://dx.doi.org/10.1609/aimag.v29i3.2157

[21] Martin Simonovsky and Nikos Komodakis. 2018. GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11139 LNCS (2 2018), 412–422. http://arxiv.org/abs/1802.03480

[22] Yorgos Tsitsikas and Evangelos E. Papalexakis. 2020. NSVD : N ormalized S ingular V alue D eviation Reveals Number of Latent Factors in Tensor Decomposition . In *Proceedings of the 2020 SIAM International Conference on Data Mining.* Society for Industrial and Applied Mathematics, 667–675. DOI: http://dx.doi.org/10.1137/1.9781611976236.75

[23] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2017. GraphGAN: Graph Representation Learning with Generative Adversarial Nets. (11 2017). http://arxiv.org/abs/1711.08267

[24] Stanley Wasserman and Philippa Pattison. 1996. Logit models and logistic regressions for social networks: I. An introduction to Markov graphs andp. *Psychometrika* 61, 3 (9 1996), 401–425. DOI: http://dx.doi.org/10.1007/BF02294547

[25] Xiang Wei, Boqing Gong, Zixia Liu, Wei Lu, and Liqiang Wang. 2018. Improving the Improved Training of Wasserstein GANs: A Consistency Term and Its Dual Effect. (3 2018). http://arxiv.org/abs/1803.01541

[26] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. (2 2018). http://arxiv.org/abs/1802.08773