GraphRNN Revisited: An Ablation Study and Extensions for Directed Acyclic Graphs

Maya Ravichandran*
maya.rav@outlook.com

Mark Koch*

Taniya Das*
Eindhoven University of Technology
5612 AE Eindhoven, Netherlands
t.das@tue.nl

Nikhil Khatri*
Quantinuum
17 Beaumont Street, Oxford OX1 2NA, United Kingdom
nikhil.khatri@quantinuum.com

.

Abstract

GraphRNN is a deep learning-based architecture proposed by You et al. [1] for learning generative models for graphs. We replicate the results of You et al. using a reproduced implementation of the GraphRNN architecture and evaluate this against baseline models using new metrics. Through an ablation study, we find that the BFS traversal suggested by You et al. to collapse representations of isomorphic graphs contributes significantly to model performance. Additionally, we extend GraphRNN to generate directed acyclic graphs by replacing the BFS traversal with a topological sort. We demonstrate that this method improves significantly over a directed-multiclass variant of GraphRNN on a real-world dataset.

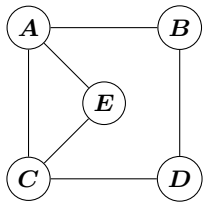
1 Introduction

The paper *GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models* by You et al. [1] proposes a deep auto-regressive model for graph generation. They present the GraphRNN (Graph Recurrent Neural Network) architecture, which can generate variable-sized graphs using a hierarchy of two RNNs. Their approach scales to graphs with large node counts by employing a breadth first search (BFS) ordering scheme that reduces the number of representations a given graph can have. You et al. evaluate their method on a suite of synthetic and real-world graph datasets, introducing new metrics for quantitative evaluation based on the Maximum Mean Discrepancy (MMD) of various graph statistics.

We demonstrate that we have a performant implementation of the original GraphRNN model by reproducing the qualitative results and reported MMD scores for the datasets used by You et al. This serves as both a reproducibility result and as a base for our extensions and ablation studies. We focus

*All authors contributed equally to this work. Listing order is random. Work done while the authors were at the Department of Computer Science, University of Oxford

NeurIPS 2023 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2023).



		S_2^π	S_3^π	S_4^π	S_5^π
A	0	1	1	0	1
B	1	0	0	1	0
C	1	0	0	1	1
D	0	1	1	0	0
E	1	0	1	0	0
	A	B	C	D	E

Figure 1: A graph and its adjacency matrix under the node ordering A, B, C, D, E. The highlighted sequence of upper triangular columns $S^\pi = (S_2^\pi, S_3^\pi, S_4^\pi, S_5^\pi)$ is enough to fully characterise the graph.

on reproducing results on the smaller versions of the datasets due to limited compute resources. The code for our reproduced implementation, evaluation, and extensions is provided[1].

Overview We begin with a brief summary of the GraphRNN architecture in Section 2. Section 3 features the results of the reproduced experiments, followed by a discussion of alternative evaluation metrics in Section 4. We run an ablation study in Section 5 to empirically investigate the importance of the BFS ordering for model performance. Further, we present a novel extension for generating directed acyclic graphs (DAGs) and report improvements over an implementation of GraphRNN for multiclass directed graphs in Section 6. Finally, we conclude by discussing related work in Section 7 and our overall results in Section 8.

2 GraphRNN summary

We briefly summarise the parts of the GraphRNN paper relevant to our experiments.

2.1 Representing Graphs as Sequences

You et al. consider undirected graphs $G = (V, E)$ given by a set of nodes $V = \{v_1, ..., v_n\}$ and a symmetric edge relation $E \subseteq V \times V$. Further, E is assumed to be irreflexive, i.e. nodes do not have self-loops.

A typical representation of graphs are *adjacency matrices*. You et al. consider adjacency matrices for all possible permutations of nodes. Given a permutation function $\pi : V \to V$, they define the adjacency matrix $A^\pi \in \{0, 1\}^{n \times n}$ as

$$A^\pi_{i,j} = \begin{cases} 1 & \text{if } (\pi(v_i), \pi(v_j)) \in E \\ 0 & \text{otherwise.} \end{cases}$$

The autoregressive model presented by You et al. generates graphs by generating their adjacency matrices column by column. More precisely, it suffices to generate the upper triangle of A^π since A^π is symmetric for undirected graphs. This is illustrated in Figure 1. Formally, undirected graphs are represented by a sequence

$$S^\pi = (S_2^\pi, ..., S_n^\pi)$$

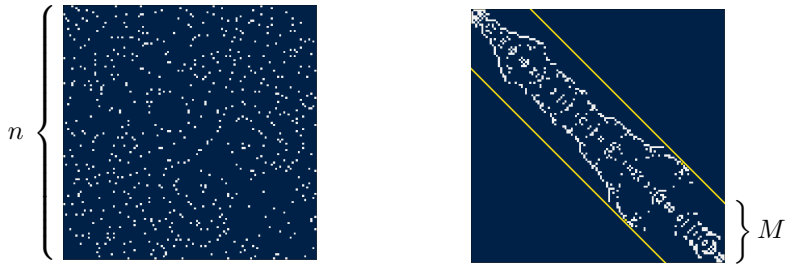
where $S_i^\pi \in \{0, 1\}^{i-1}$ is the i-th column of the upper right triangle of A^π, i.e.

$$S_i^\pi = (A^\pi_{1,i}, ..., A^\pi_{i-1,i})^T.$$

Instead of learning to generate graph sequences S^π for all possible permutations π, You et al. only consider node orderings that arise from breadth first search (BFS) traversals of the graph. They claim that this approach has two main benefits:

- Since many graphs have less than $n!$ BFS orderings, this technique usually reduces the number of different representations a graph can have.

[1] https://github.com/maravichandran/GraphRNN



(a) (b)

Figure 2: (a) Shows the adjacency matrix of a random graph under a random node ordering. (b) Depicts the adjacency matrix of the same graph under a BFS ordering. We can see that the entries are closer to the diagonal under the BFS ordering, allowing us to represent the upper triangle using smaller slices of length M.

- If π is a BFS ordering, then for all $i < j$, $A^{\pi}_{i,j-1} = 1$ and $A^{\pi}_{i,j} = 0$ implies that $A^{\pi}_{i',j'} = 0$ for all $i' \leq i$ and $j' \geq j$ (see Proposition 1 in [1]). As a consequence, the entries of A^{π} are concentrated around the diagonal of the matrix as shown in Figure 2. This allows us to shrink the size of the vector S^{π}_i needed to represent each column to some fixed length $M \leq n$ that only includes the entries around the diagonal as illustrated in Figure 2b. Formally,

  $$S^{\pi}_i = (A^{\pi}_{\max(1,i-M)}, ..., A^{\pi}_{i-1,i})^T.$$

  This reduces the number of predictions the generation model needs to make.

You et al. highlight this BFS technique as a crucial component of their approach and claim that it drastically improves scalability. We empirically investigate the contribution of BFS in Section 5.

2.2 GraphRNN Architecture

You et al. propose a hierarchy of two recurrent neural networks (RNNs) to autoregressively generate S^{π}. The first RNN operates on the node level. Given a sequence $S^{\pi}_1, ..., S^{\pi}_i$ of previously generated adjacency vectors, it maintains a hidden representation h_i that captures the global state of the graph. This hidden state is used to initialise the second RNN that operates on the edge level and is used to autoregressively sample the components of S^{π}_{i+1}, i.e. the connectivity of the $i+1$ 'th node. You et al. also introduce a simplified version of their model called GraphRNN-S in which the second edge-level RNN is replaced with a multi-layer perceptron (MLP) generating a vector θ_{i+1} of probabilities from which S^{π}_{i+1} is sampled independently.

2.3 Evaluation Metrics

Quantitatively evaluating the quality of a generated graph is a challenging task. Graph-matching is in general intractable, and thus You et al. suggest quantifying the quality of generated graphs through comparison of a set of graph statistics. To compare node-level characteristics, such as degree and clustering coefficient between two graphs, they suggest the use of maximum mean discrepancy (MMD) to measure the similarity of their distributions. You et al. use the MMD measure on three metrics: degree, clustering coefficients, and orbit statistics [2] distributions.

2.4 Experiments

Datasets You et al. evaluate their models on two synthetic and two real-world graph datasets. The synthetic ones include a *community* dataset (two-community graphs with $60 \leq |V| \leq 160$ generated using the Erdős-Rényi model (ER) [3]) and a *grid* dataset (2D rectangular grid graphs with side length ranging from 10 to 20). The Citeseer network [4] is used to generate 3-hop graphs with $50 \leq |V| \leq 399$, forming the *ego* dataset. Finally, You et al. use a *protein* dataset consisting of graphs with $100 \leq |V| \leq 500$ representing protein structures. In addition, they evaluate smaller equivalents of these datasets.

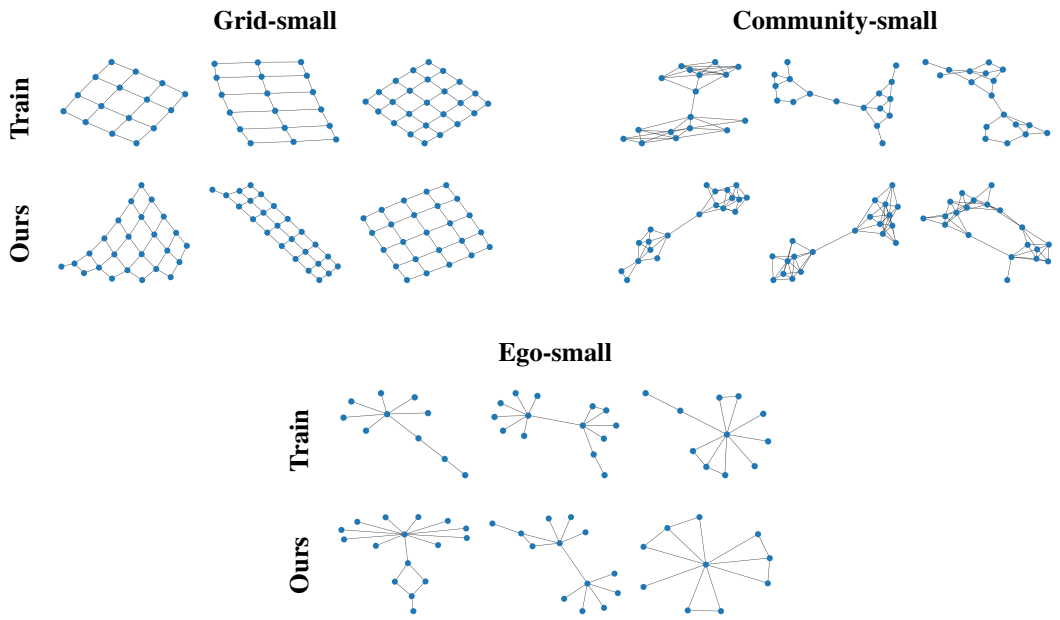
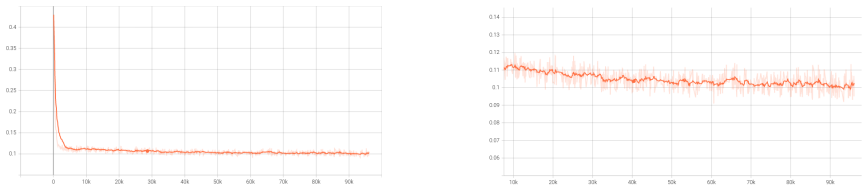


Figure 3: Generated graphs from our implementation of GraphRNN compared to the training data.



(a) (b)

Figure 4: Loss curve for training GraphRNN on the *community-small* dataset. Part (a) shows the full curve and part (b) the curve after 10k iterations.

Baselines You et al. compare their methods against the rule-based Erdős-Rényi (ER) [3] and Barabási-Albert (BA) [5] models that generate graphs through random processes. They also compare their methods against Kronecker graph models [6], mixed-membership stochastic block models [7], and the deep learning approaches GraphVAE [8] and DeepGMG [9].

3 Reproduced Experiments

For our reproducibility experiments, we focus on the scaled-down datasets *community-small* and *ego-small*. Further, we generated our own *grid-small* dataset containing smaller grids of size up to 6×6. Since grid-small is not part of the original paper, we applied the BA and ER baselines to show a similar performance improvement for this new dataset. We used smaller datasets than those in the original paper due to having limited compute resources.

We implemented the GraphRNN architecture in PyTorch following the instructions in [1]. In particular, we use the configuration for the smaller model mentioned in Appendix A.1 to match the small datasets and to speed up training. In dataset generation, when there were discrepancies between You et al.'s paper and code, we use the parameters mentioned in the paper. For the MMD measure and orbit metric, we use You et al.'s implementation.

We use a random 80%/20% train/test split and train the models for 96k iterations with a batch size of 32, as mentioned in [1]. However, we notice that training usually converges after 10k-30k iterations, which is likely due to the smaller size of the datasets and the smaller GraphRNN configuration used. Across our experiments, we thus use checkpoints from the first 30k iterations of training. Figure 4

Table 1: Comparison of our GraphRNN (GRNN) implementation with the original implementation and baselines (BLs), using the MMD measures of degree, clustering and orbit distributions

		community-small			ego-small			grid-small		
		deg.	**clus.**	**orbit**	**deg.**	**clus.**	**orbit**	**deg.**	**clus.**	**orbit**
BLs	**BA**	0.459	0.707	0.083	0.159	0.589	0.012	1.206	**0**	0.084
	ER	**0.016**	0.193	0.012	0.136	0.050	0.035	0.242	0.656	0.024
[1]	**GRNN-S**	0.020	0.150	0.010	0.002	0.050	**0.001**	NA	NA	NA
	GRNN	0.030	0.030	0.010	**3e-4**	0.050	**0.001**	NA	NA	NA
Ours	**GRNN-S**	0.038	**0.007**	0.006	0.004	**0.032**	**0.001**	0.080	0.065	0.007
	GRNN	0.031	0.012	**0.001**	0.004	0.038	0.003	**0.065**	**0**	**0.001**

Table 2: Comparison between our implementation of GraphRNN (GRNN) and the BA and ER baselines. We use the MMD of betweeness centrality (BC) and closeness centrality (CC), as well as the absolute difference of density (D) and transitivity (T).

		BA	**ER**	**GRNN-S**	**GRNN**
community-small	**BC**	0.054	0.038	**0.024**	**0.024**
	CC	0.026	0.022	**0.021**	0.022
	D	0.060	0.055	0.045	**0.044**
	T	0.297	0.205	0.167	**0.166**
ego-small	**BC**	0.092	0.065	**0.062**	0.085
	CC	0.058	0.049	**0.045**	0.054
	D	0.049	0.075	0.068	**0.039**
	T	0.197	0.189	0.172	**0.146**
grid-small	**BC**	0.436	**0.409**	0.430	0.470
	CC	0.393	**0.368**	0.384	0.381
	D	0.082	0.026	0.021	**0.020**
	T	0	0.256	0.016	**0**

shows a typical loss curve observed during training. Note that we applied an accelerated learning rate decay schedule compared to [1] due to using smaller model configurations.

3.1 Results

Figure 3 visualises graphs generated using our implementation of GraphRNN compared to graphs from the training set. This shows that our implementation is capable of generating the structure of grid, community, and ego graphs, matching the qualitative results presented in Figure 2 in the original paper. Similar to You et al., we also observe that the model generates grids that are not present in the training data, for example, the 3×8 grid shown in Figure 3.

Table 1 presents a quantitative comparison of our implementation of GraphRNN with You et al.'s results and baseline results for the MMD scores. We generate baseline metrics using the BA and ER models, since the GraphRNN paper does not provide these baselines for the smaller datasets. Overall, we observe similar improvements over the baselines as reported in [1].

4 Alternate Graph Similarity Statistics

The graph similarity metrics considered by You et al. [1] (degree distribution, clustering distribution, and orbit statistics) are primarily concerned with individual node statistics and local structures in graphs. To provide further quantitative insight into the properties of graphs generated by GraphRNN, we consider some additional metrics, described below.

Betweenness Centrality (BC) Betweenness centrality for a node is the proportion of shortest paths in the graph that pass through that node. Compared to degree, clustering, and orbit statistics, which mainly focus on local graph structures, BC provides a view of the global connectivity of graphs. Following the previous experiments, we make use of MMD to compare the BC distributions of different graphs.

Table 3: M values for considered datasets, together with the reduction they represent from the maximum number of nodes in a graph in each dataset

Dataset	Min nodes	Max nodes	M	Reduction
ego-small	4	18	16	11.11%
community-small	12	20	16	20%
grid-small	4	36	12	66.67%

Closeness Centrality (CC) Closeness centrality is a node-level measure defined as the reciprocal of the sum of shortest paths to all other nodes in the graph. Similar to BC, CC captures global graph connectivity. Again, we compare distributions via MMD.

Density (D) Density is graph-level measure defined as the ratio between the number of edges present in the graph and the number of possible edges in the graph if each node in the graph were connected to all other nodes. We compare the mean absolute density difference between generated graphs and the graphs in the test set.

Transitivity (T) Transitivity is a graph-level property defined as

$$t = \frac{3 \times \text{\#triangles}}{\text{\#triads}},$$

where triads are 3 nodes sharing at least 2 edges. Like the clustering coefficient, transitivity is a measure of the graph property of triadic closure. Unlike the clustering coefficient that only describes how densely single nodes are connected, transitivity measures how transitive the edge relation E is. Similar to density, we evaluate the mean absolute difference of transitivity between generated graphs and the test set.

4.1 Results

Table 2 presents a comparison of our GraphRNN implementation with baseline models for the alternative statistics described above. For the community-small dataset, we observe that both GraphRNN-S and GraphRNN outperform the baselines for all new metrics. We observe that there is negligible difference in the performance of GraphRNN-S and GraphRNN. This is consistent with the results of Table 1, where both models perform similarly. For the ego-small dataset, the best score for each metric is achieved by the GraphRNN or GraphRNN-S models. For the grid-small dataset, we observe that the ER model achieves slightly lower values for the centrality measures. The BA model is able to perfectly replicate the transitivity of the grid-small dataset. This is also consistent with BA achieving a perfect score for clustering coefficient (another measure of triadic closure) on the grid-small dataset in Table 1. In both cases, our implementation of GraphRNN is able to match this result.

5 Ablation - Investigating the Contribution of BFS

We perform an ablation study to test You et al.'s claims regarding the benefit of BFS ordering of the graph adjacency matrix. The purpose of this study is to determine the extent to which the reduction in the number of graph representations and the reduced number of edges that the model has to predict improve the performance. The following experiments are performed:

GraphRNN-no-M This experiment aims to assess the effect of reducing the number of edges the model has to predict on performance. Here, the nodes are arranged in BFS order as in the default GraphRNN model, but the input sequence is not truncated to be of size M.

GraphRNN-no-BFS: This experiment aims to assess the effect of reducing the number of graph representations on performance. Here, the adjacency matrix for each graph is permuted randomly, and the vector sequence $S^\pi = (S_2^\pi, ..., S_n^\pi)$ is fed into the model without further processing.[2]

Both these experiments are run for the *grid-small*, *ego-small* and *community-small* datasets. Table 3 lists the minimum and maximum number of nodes in each considered dataset, together with the

[2]Note that since a BFS traversal is not used for π, an M value cannot be calculated.

Table 4: Comparison of the GraphRNN model with the GraphRNN-no-M and GraphRNN-no-M ablation models using the MMD measures of degree, clustering and orbit distributions

		community-small			ego-small			grid-small		
BFS	**M**	**deg.**	**clus.**	**orbit**	**deg.**	**clus.**	**orbit**	**deg.**	**clus.**	**orbit**
✓	✓	**0.031**	**0.012**	**0.001**	**0.004**	0.038	0.003	**0.065**	**0.000**	**0.001**
✓	×	**0.031**	0.027	**0.001**	0.008	**0.023**	**0.001**	0.073	0.126	0.018
×	×	0.034	0.107	0.023	0.067	0.096	0.022	0.134	0.660	0.018

M values calculated by considering 9000 random BFS permutations of graphs from each. The grid-small dataset has a particularly low value of M (which we attribute to the low density and node degrees of grid graphs), yielding a significant reduction in the number of edges the model needs to predict. In comparison, the ego-small and community-small datasets have M values much closer to the maximum number of nodes.

We make the following hypotheses about the ablation study:

Hypothesis 1: In graphs where the M value is close to the maximum number of nodes, not much performance is gained through the use of M-reduced data.

Hypothesis 2: BFS traversal-based ordering contributes significantly to performance in all cases.

Hypothesis 1 is suggested by the fact that when M is close to the number of nodes in a graph, it does not reduce the number of edge predictions required from the model by an appreciable amount. Thus, the model's performance should, at best, be only mildly improved. *Hypothesis 2* is motivated by the fact that there are typically far fewer than $n!$ BFS traversals of a graph (an argument also used by You et al. in favor of the BFS traversal). These hypotheses are investigated experimentally below.

5.1 Ablation Results

The results of our ablation experiments are shown in Table 4. For the community-small dataset, we observe that there is no decrease in performance on removing the M-reduction from the model for the degree and orbit metrics. There is a slight decrease in performance as measured along the clustering MMD metric. This is consistent with *Hypothesis 1*, since the M value for community-small is only 20% less than the maximum number of nodes in a graph for this dataset, as shown in Table 3. On removing BFS from the model, a substantial decrease in performance is visible. The degree metric is slightly worse, while the clustering and orbit metrics are worse by an order of magnitude.

In the case of ego-small, on removing M, the model exhibits slightly better results for clustering and orbit, and slightly worse results for degree. Since the M value reduction for this dataset is only 11%, these results are also consistent with *Hypothesis 1*. On removing BFS, we observe a consistent decrease in performance across all 3 metrics. Degree and orbit metrics worsen by an order of magnitude.

For the grid dataset, removing M substantially worsens clustering and orbit MMD scores and mildly worsens the degree MMD score. Observe that for grid, the M value reduction is significant, at 67%. Thus, the observed sizeable regression in performance provides evidence to support *Hypothesis 1*. Removing BFS from the model further worsens performance for degree and clustering metrics, while not impacting orbit.

Hence, the data suggests that *Hypothesis 1* may be true, as the removal of M significantly impacts only the datasets with high M-reduction percentages. Across datasets, we observe that removing BFS significantly hampers performance. This provides evidence in support of *Hypothesis 2*.

6 Directed Graphs

The original GraphRNN model considers only the case of undirected graphs. However, many graphs in practical applications are directed. In a directed graph, the edge relation E need not be symmetric. Under this relaxation of E, the adjacency matrix A is no longer necessarily symmetric. Note that

we continue to assume irreflexivity. We consider two avenues of extending GraphRNN to generate various classes of directed graphs.

6.1 Generating Arbitrary Directed Graphs

So far, the graph sequences S^π used to model graphs only contained zeros and ones, indicating whether nodes are connected or not. In order to encode directed graphs, we introduce the notion of a directed graph sequence $S^{\to\pi}$ that also captures the direction of edges:

$$S^{\to\pi} = (S_i^{\to\pi}, ..., S_n^{\to\pi})$$

where

$$S_i^{\to\pi} = (A_{1,i}^{\to\pi}, ..., A_{i-1,i}^{\to\pi})^T \quad \text{for } 2 \leq i \leq n,$$

$$A_{i,j}^{\to\pi} = \begin{cases} (1,0,0,0) & \text{if } (\pi_i, \pi_j) \notin E \wedge (\pi_j, \pi_i) \notin E, \\ (0,1,0,0) & \text{if } (\pi_i, \pi_j) \in E \wedge (\pi_j, \pi_i) \notin E, \\ (0,0,1,0) & \text{if } (\pi_i, \pi_j) \notin E \wedge (\pi_j, \pi_i) \in E, \\ (0,0,0,1) & \text{if } (\pi_i, \pi_j) \in E \wedge (\pi_j, \pi_i) \in E. \end{cases}$$

You et al. mention directed graphs as a special instance of graphs with edge-features. Specifically, each edge belongs to one of two classes, indicating its direction. We use a slight modification of this proposal. In our implementation, we encode the edge class as a one-hot vector of length 4. We use a softmax activation as the final layer of our RNN-based edge-level model to generate a probability distribution over edge classes. This extension of GraphRNN, which represents directed graphs through multiple edge classes, is called *GraphRNN-DIR*. Nodes are ordered by a BFS traversal of the graph which respects edge directions.

6.2 Generating Directed Acyclic Graphs

While the above modification to GraphRNN is suitable for the general case of directed graphs, several interesting graphs exist with known structural constraints. For example, several graphs of interest are known to be directed acyclic graphs (DAGs). We present a novel extension of GraphRNN to the special case of generating DAGs.

Recall that the input sequence to the GraphRNN model S^π consists of the upper right triangular component of the graph's adjacency matrix. We can formalise the notion of upper triangular matrices as follows:

Definition 1. *A matrix A is upper right triangular if $A_{i,j} = 0$ for all i, j with $i \geq j$.*

It turns out there is an interesting connection between DAGs and triangular matrices:

Theorem 1. *An adjacency matrix A^π is upper right triangular iff π is a topological ordering of the associated graph.*

Proof. We prove both directions separately:

$\Rightarrow$: Suppose A^π is an upper triangular adjacency matrix of some graph G under some ordering π. Let $j < i$. We have to show that there is no edge from $\pi(v_i)$ to $\pi(v_j)$. This trivially holds since $A^\pi_{i,j} = 0$ as A^π is upper triangular. Thus, π is a topological sort.

$\Leftarrow$: Suppose the ordering π represents a topological sort. We claim that A^π is an upper triangular matrix. Let $i \geq j$. Then, $\pi(v_i)$ is the i-th node in the topological sort, and $\pi(v_j)$ is the j-th node in the topological sort. Since $i \geq j$, there cannot be an edge from the i-th node to the j-th node (otherwise the i-th node would need to appear before the j-th node in π, giving rise to a contradiction). Therefore, we must have $A^\pi_{i,j} = 0$ such that A^π is upper triangular.

□

Thus, we can describe DAGs using the original sequence representation S^π if π is a topological sort. We propose to replace the previously used BFS ordering with a topological ordering of the nodes and then apply the original GraphRNN model to learn the distribution of those sequences. Interpreting

Table 5: Comparison of GraphRNN-DIR and GraphRNN-DAG models for the *ego-directed* dataset, using MMD measures of degree and clustering

	Degree	**Clustering**	**Components / graph**
GRNN-DIR	0.746	1.343	**1**
GRNN-DAG	**0.054**	**0.125**	1.3

the generated sequences as an upper triangular matrix ensures that all graphs generated by the model are DAGs. We call this architecture GraphRNN-DAG.

A downside of this method is that we are no longer able to prune the dimension of the adjacency vectors S_i^π to a smaller size M, since the topological sort does not reduce the number of necessary edge-predictions in the manner BFS does.

6.3 Evaluation

To evaluate our topological sort approach's performance, we consider a modification of the ego-small dataset. This is a natural choice since citation graphs are largely acyclic and have a direction associated with each edge (indicating either a "cites" or a "cited-by" relation). We take the largest weakly-connected component of the citeseer dataset [10] and generate 200 directed 3-hop[3] ego networks centering on random nodes in this graph. This dataset has $7 \leq |V| \leq 30$, and an M value of 25. This data is then passed to 2 models: *GraphRNN-DIR* and *GraphRNN-DAG*, described in the preceding sections. The graphs generated from these trained models were evaluated using MMD comparison of degree and clustering metrics with the test set. We omit the comparison of orbit MMD since the implementation is limited to undirected graphs.

As shown in Table 5, the GraphRNN-DAG model outperforms the multiclass approach (GraphRNN-DIR) along each MMD metric by a significant margin. One weakness of the topological sort model is that it occasionally generates graphs with multiple weakly connected components. This does not mirror the training dataset, where each graph is a single weakly connected component.

7 Related Work

Several works that either build upon the GraphRNN model or propose alternative frameworks for graph generation exist. For example, [11] proposes the usage of a depth-first search (DFS)-based node ordering within the GraphRNN model, finding that the DFS node ordering yields similar performance results to the BFS node ordering. In addition, the Graph Recurrent Attention Networks (GRAN) model [12] builds upon the GraphRNN model by incorporating an attention mechanism to capture conditional dependencies between existing and new graph structures. Other approaches include diffusion and transformer-based models for graph generation, such as DiGress [13], a denoising diffusion-based transformer model for graph generation.

8 Discussion

In this work, we implemented the GraphRNN model proposed by You et al. [1] and reproduced their numerical results for smaller datasets. We evaluated this implementation against BA and ER baselines using new metrics, finding improvements across the board. The results of our ablation study in Section 5 show that the BFS traversal contributes significantly to model performance across datasets. We extended GraphRNN to directed graphs by introducing multiclass predictions. We further presented a novel extension of GraphRNN to directed acyclic graphs and evaluated this against the GraphRNN-DIR implementation. A key insight from our work is that using a topological sort in place of a BFS traversal yields improved performance for generating DAGs. Further experiments are necessary to see if our results hold for larger dataset sizes. An avenue for future research is to identify other domain-aware traversals which may yield improved isomorphism-reductions for specific datasets.

[3]The hops ignore edge direction.

References

[1] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018.

[2] Tomaž Hočevar and Janez Demšar. A combinatorial approach to graphlet counting. *Bioinformatics*, 30:559–565, 2014.

[3] Paul Erdős, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[4] 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.

[5] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[6] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: an approach to modeling networks. *Journal of Machine Learning Research*, 11(2), 2010.

[7] Edo M Airoldi, David Blei, Stephen Fienberg, and Eric Xing. Mixed membership stochastic blockmodels. *Advances in neural information processing systems*, 21, 2008.

[8] Martin Simonovsky and Nikos Komodakis. GraphVAE: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*, pages 412–422. Springer, 2018.

[9] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[10] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pages 89–98, 1998.

[11] Ousmane Touat, Julian Stier, Pierre-Edouard Portier, and Michael Granitzer. GRAN is superior to GraphRNN: node orderings, kernel- and graph embeddings-based metrics for graph generators, 2023.

[12] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Charlie Nash, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard S. Zemel. Efficient graph generation with graph recurrent attention networks, 2020.

[13] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete denoising diffusion for graph generation, 2023.