CS224W: Project Final Report
Deep Generative Models for Graph Generation

Gorish Aggarwal, Susanna Maria Baby, Akshat Jindal*
{gorish,susmaria,ajindal}@stanford.edu

ABSTRACT

Deep generative models for graph generation is a very popular topic currently in the deep learning community and several successful approaches have been developed in recent years. Our team aims to do a replication and comparative study of various Generative Adversarial Models [6] for generating both real world graphs and other interesting graphs: We try to find how each of these generative models differ fundamentally from each other, and how they perform on various metrics like extensibility and efficiency in addition to traditional metrics like degree distribution, clustering coefficient, features of largest connected component etc on different types of input training data. We will be training these models on different types of datasets like grid dataset, enzymes data, community data, large real world networks and other new, interesting datasets to further capture the nuance of each model.

1 INTRODUCTION

Graphs have become a quintessential form of data storage and analysis across a plethora of domains. Relational data in the fields of social networks, medicine (drug discovery), population topologies etc are represented as graphs. Thus the process of graph generation has some very important applications like discovering novel structures like drug discovery, graph completions, prediction of future links based on the current graph etc. We studied graph generation in class across multiple lectures first by using random models like Erdős–Rényi [4], Small-World [13] etc and then an in depth discussion on a very famous deep generative approach: GraphRNNs. This motivates us to explore this topic in detail.

Generating graphs through deep generative models is an inherently difficult problem due to several reasons : Unlike images that deal with real valued data, graphs and texts are discrete objects - adapting GANs for working with such data is still an open problem. There is a lack of large repositories of graphs that come from the same underlying distribution. So typical learning involves training on a single graph. Further, the generated graph has to be permutation-invariant (non-isomorphic) to the original graph, while retaining all of its interesting properties. Despite these challenges, several promising models for graph generation using deep learning have emerged recently. In this project, we will be exploring some of these models in more detail and see how they compare against each other.

Our project aims to conduct methodological and empirical research on three different generative graph models: NetGAN [2], Graph Recurrent Attention Network (GRAN) [11] and GraphRNN [14] using common metrics to understand the performance of these generative graph models on various real-world datasets, as well as characteristics such as scalability, ability to handle complex dependencies and non-unique representations, generalization ability, and robustness. Through this study, we also hope to find the most compelling factors that contribute to the success of each of these generative models. As far as we know, there is no comprehensive comparison study between the models we discuss together till date.

We will then test these models on new datasets (synthetic and real world) not used in the original papers and analyze the difference in structures each model is capable of generating. It is interesting to see how NetGAN which is fundamentally different from the other two models compares with each other.

2 PRIOR WORK

Some of the simplest models that are often used as baselines for graph generation techniques are Erdős–Rényi model [4] for random graph generation, Watts–Strogatz model [13] for generating graphs with small world properties and Kronecker graph model [10] for constructing large realistic graphs recursively from smaller ones. While these explicit models are successful in capturing some properties of real world graphs, they fails terribly on others. A possible solution to graph generation that captures interesting properties is to move to implicit models like GANs [6], which have been very successful in several fields like image and text generation. However, unlike image and text which are very ordered structures, graphs have no explicit ordering, which makes graph generation using GANs a difficult task. Nevertheless, several deep generative models have shown promise in this direction, and we will be exploring three of them in more detail in the next sections.

*All members contributed equally: Gorish worked on Graph RNNs, Susanna worked on NetGAN and Akshat worked on GRAN model

3 MODELS FOR COMPARISON

We now present the 3 graph generative models in enough detail to make the reader aware of what's going on in each model and encourage the reader to go into the respective original papers as cited for more details. We will stress upon the required details of the models further below when we analyze the differences between them.

NetGAN: Generating Graphs via Random Walks [2]

In this paper, the authors introduce NetGAN [2] : a new novel graph generative model. Here, the problem of learning the graph topology is formulated as as learning the distribution of biased second order random walks [7] over the graph. The generator G of the GAN is modeled as a stochastic neural network with discrete outputs. G learns to generate random walks that are plausible in the real graph using Wasserstein GAN (WGAN) framework [1], while the discriminator D then has to distinguish them from the true ones that are sampled from the original graph. Some of the salient properties of this GAN architecture are listed below :

- Even though NetGAN does not explicitly specify any particular properties (like degree distribution, clustering coefficient, characteristic path length etc), the model is able to achieve distributions similar to the original graph in the generated graphs consistently, without trivially reproducing the same input graph.
- The model has also been shown to have competent performance on link prediction - another task that was not explicitly modelled which once again emphasizes the generalization property of the model.
- NetGAN also claims high efficiency and scalability. Since it operates using only random walks - NetGAN exploits the efficiency of sparsity of adjacency matrices of real world graphs - thereby running in almost linear time in the number of nodes.
- Random walks are also invariant to node reordering which is another preferred attribute while dealing with graphs.
- NetGAN uses early stopping through Val-Criterion (when validation performance stops improving for a certain number of iterations) or through EO-Criterion (Edge Overlap) to prevent the generator from memorizing the input graph.
- Straight-Through Gumbel estimator [8] is used to estimate sampling from categorical distribution for sampling nodes in the random walks.
- Latent space interpolation reveals smoothly changing properties in the generated graphs suggesting that NetGAN learns to map specific parts of the latent space to specific properties of the graph.

GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models [14]

The paper proposes a deep autoregressive model, GraphRNN [14] to address the challenge of modelling complex distribution of real-world graphs and approximate any distribution of such graphs with minimalist assumptions about their structure. It does so by modeling the graphs in an recurrent manner—as a sequence of additions of new nodes and edges—to capture the complex joint probability of all nodes and edges in the graph. The overall framework decomposes the process of graph generation into 2 RNNs, first being the a graph-level RNN to generate sequence of nodes and second, an edge-level RNN to generate the sequence of edges for each new node. The salient features of the model include:

- GraphRNN learns the distribution over edge connections for i-th node conditioned on previous nodes' edge connections parameterized with an expressive neural network
- The neural network could be varied to correspond to different assumptions about $p(S_i^\pi | S_{<i}^\pi)$. For instance, it could be a Multivariate Bernoulli or a Dependent Bernoulli sequence. We use the Dependent Bernoulli scheme.
- The model includes a Breadth-First search (BFS) node-ordering scheme, which improves the scalability by reducing the overall number of sequences to consider as well as reducing the number of edge of predictions.
- The paper also provides metrics for comprehensive evaluation, which are based on Maximum Mean Discrepancy (MMD) to measure the distance between graphs, which compares higher order moments of graph-statistics distributions.

Efficient Graph Generation with Graph Recurrent Attention Networks [11]

In this paper, the authors add to the growing list of deep generative models for graphs by introducing Graph Recurrent Attention Networks or GRANs[11]. This auto-regressive model essentially generates one block of nodes and edges at a time conditioned on the sub-graph already generated by using GNNs to get updates node representations at every time step and learn the probability distribution over each new potential edge as a mixture of Bernoulli distributions. To overcome the problem of having an exponential number of permutations of nodes to take care of, the authors use a *canonical ordering* of nodes to restrict the permutation set they consider while estimating likelihood of the graph. Being the latest paper claiming state of the art performance in graph generation, the authors claim to resolve issues with older models in the following major ways:

- By generating a block of nodes at every auto regressive step, the authors are able to speed up the generation process by reducing the number of time steps taken to generate the entire graph. More specifically, GRANs use $O(V)$ steps whereas the previous best method, GraphRNN[14] used $O(V^2)$ steps.
- By using an attention based GNN, they are able to better use the topology of the already generated graph for estimating the distributions of the new edges at each time step.
- To ensure that the newly added edges are not independent of each other, a mixture of Bernoullis is used at each time step to capture the correlations between the new edges generated.
- The set of canonical orderings used provides an efficient solution for handling node permutations.
- The authors claim and experimentally show the scalability of their approach by using the *Point Cloud* dataset with upto 5k nodes. May famous previous approaches like GraphVAE and GraphRNN are unable to scale to this dataset but GRANs does well!
- The authors used not only local metrics like degree distributions, clustering coefficient distributions, and the number of occurrence of all orbits with 4 nodes for evaluation but also compared the spectral clustering of the real and generated graph for a global metric.

4 DATASETS

The datasets for our current models can handle are mid sized graph collections with nodes varying between 10 and 1000 nodes. For the experiments, we are training our models on 3 synthetic and 3 real world datasets.

- **Grid Dataset**: The grid dataset was constructed using the networkX library. We created 100, 2-d grid graphs with the height and width of the graphs varying from (10,10) to (19,19) in increasing steps of 1.
- **Smaller Grid**: The smaller grid dataset was constructed using the networkX library. We created 100, 2-D grid graphs with height and width varying from (2,5) to (2,6) in increasing steps of 1.
- **Triangular**: The triangle dataset was constructed the networkX library. 100, 2-d graphs with planar, triangular structure having heights and widths varying from(10,10) to (20, 20) in increasing steps of 1.
- **Enzyme Dataset**: The enzyme dataset[3] consists of 918 protein graphs with number of vertices varying from 100 to 500.
- **Citeseer**: 757 3-hop ego networks extracted from the Citeseer network [5] with $50 \leq |V| \leq 399$. Nodes represent documents and edges represent citation relationships.
- **Cora**: 1480 3-hop ego networks extracted from the Cora dataset [12] with $50 \leq |V| \leq 398$. Nodes represent documents and edges represent citation relationships.

For each of them, We used 80% of the graphs as our training set and the rest 20% as our test set in the GraphRNN and the GRAN model.

5 EXPERIMENTS AND EVALUATIONS

Methods

NetGAN. NetGAN is fundamentally different from the other two models in the sense that NetGAN learns by training on a single graph, while GraphRNN and GRAN work on a collection of graphs. To work around this, for each type of dataset (grid and enzymes), we trained NetGANs on three different randomly chosen graphs from the training data, and generated several graphs from each of the 3 NetGAN models. To ensure that NetGAN does not just memorize the input graph, we employed three different regularization techniques: stop training once the edge overlap of generated graphs with respect to the original graph reaches 85%, use a fixed number of maximum training iterations, and work on a lower dimension space H for learning the logits which is then projected up to N using a projection matrix $W_{up} \in R^{H \times N}$. The original authors of NetGAN used the projection up and down trick for computational efficiency while working with large graphs like Cora [12] that has thousands of nodes. We keep $H \leq 0.66N$ for generalization purposes. Most of the other hyper-parameters like starting temperature (0.5), initial learning rate (0.0003), optimizer (Adam [9]), number of nodes in generator layer (40) and discriminator layer (30), p and q parameters tuning [7] for biased random walks etc were kept identical to the original paper.

GraphRNN. The GraphRNN model, was trained on the collection of graphs from the 6 datasets. The training and the testing data was homogenized across the models to report correctness of accuracy. Here, the testing data is used as a proxy for the actual distribution of the data and the metrics are compared with those of the generated graphs. In the GraphRNN, the graph-level RNN uses 4 layers of Gated Recurrent Unit (GRU) cells, with 128 dimensional hidden state for the larger model, and 64 dimensional hidden state for the smaller model in each layer. The edge-level RNN uses 4 layers of GRU cells, with 16 dimensional hidden state for both the larger model and the smaller model. To output the adjacency vector prediction, the edge-level RNN first maps the highest layer of the 16 dimensional hidden state to a 8 dimensional vector through a MLP with ReLU activation, then another MLP maps the vector to a scalar with sigmoid activation. The Adam Optimizer is used for minibatch training. Each minibatch contains 32 graph sequences. We train

the model for 10k epochs for Grid dataset as opposed to the 3k epochs recommended by the paper, whereas the enzyme dataset was trained for only 2200 epochs due to the time constraint. We set the learning rate to be 0.001, which is decayed by 0.3 at step 400 and 1000 in all experiments.

GRAN. The GRAN model was trained on all 6 datasets except the "Smaller Grid" dataset. We used the same split for training and testing for a fair comparison between the models. We generated 20 graphs for each dataset and compared these to 20 graphs in the test dataset for metric calculation. In fact this was kept uniform across all three models for fairness in comparison. The hidden dimensions are set to 512 for all datasets. Block size and stride are both set to 1. The number of Bernoulli mixtures is 20 as in the paper. We use a 7 layered GNNs and unroll each layer for 1 step.

The Grid dataset was trained for 10k iterations as opposed to 3k mentioned by the authors and increased the number of dimensions in the hidden layer from 128 to 512 as we were not getting decent results with the hyperparameters set as according to the authors. Thus following this, we trained the Triangular dataset for 10k iterations too with 512 hidden layers.

The three real world datasets were all trained for around 3k iterations as they took considerable longer to train and started giving good results in 3k iterations.

Evaluation Metrics

To compare these models, we will inspect :

- Visual quality of generated graphs
- Quantitative metric comparison of the generated graphs
- Training times of each model

For quantitative comparisons, we will first calculate 4 (3 local, and 1 global) MMD statistics for each of these model for each dataset[14]. The MMD statistic is a measure of distance between the distribution induced by the generated graphs and the distribution induced by the test graphs. Additionally we will also compute metrics that are characterise of the basic structure of individual graphs and report their average across all generated graphs.

(1) MMD statistics for :
   - Degree distribution **[Local]**
   - Clustering coefficient distribution **[Local]**
   - No. of occurrences of all orbits with 4 nodes (to capture higher-level motifs) **[Local]**
   - Spectra of the graph **[Global]**

(2) Gini Index to capture statistical dispersion in the degree distribution

$$G = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|d_i - d_j|}{2n\sum_{i=1}^{n} d_i} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|d_i - d_j|}{2n^2 d_{avg}}$$

(3) Number of connected components

(4) Claw count : number of occurrences of bipartite graph structure $K_{1,3}$ (star graph with one center, 3 neighbors)

$$\sum_{i=1}^{n}\binom{d_i}{3} = \frac{1}{6} \times \sum_{i}^{n} d_i(d_i - 1)(d_i - 2)$$

(5) Min degree

(6) Max degree

(7) Average degree

(8) Assortativity : Pearson correlation coefficient by degree between pairs of linked nodes. Assortativity values range from -1 to 1. It is a measure of preference for a network's nodes to attach to other nodes of similar degree.

(9) Wedge count [number of paths of length 2] : calculated using degree distribution as

$$\sum_{i=1}^{n}\binom{d_i}{2} = \frac{1}{2} \times \sum_{i}^{n} d_i(d_i - 1)$$

(10) LCC size : Size of Largest Connected Component

(11) Triangle Count

(12) Relative edge distribution entropy is a measure of the equality of the degree distribution. It is one when all degrees are equal, and attains the limit value of zero when all edges attach to a single node.

$$H_{er} = \frac{1}{ln(n)}\sum_{i=1}^{n} -\frac{d_i}{2m} ln(\frac{d_i}{2m}) \text{ where } m = \frac{1}{2}\sum_{i}^{n} d_i$$

(13) CPL : Characteristic Path Length (Average shortest path length)

(14) Clustering coefficient :

$$C = \frac{1}{N}\sum_{i=1}^{n} C_i = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{|\{e_{jk} \in E : e_{ij} \in E \wedge e_{ik} \in E\}|}{d_i(d_i - 1)}\right)$$

(15) PowerLaw exponent of degree distribution

6 RESULTS

Training Time

- NETGAN : Each NetGAN model took 2-4 hours for training on a K-80 machine.
- GRAN : The model took around 12 hours on a K-80 to train the Grid and Triangular dataset for 10k iterations and we stopped training the model on the real-world datasets after 15-16 hours allowing it to run for around 3k iterations.
- GraphRNN : The training time was 15 hours for 2200 epochs on the real-world datasets and 9 hours for 10k epochs on the synthetic datasets using a Tesla-V100 GPU.

Qualitative and Quantitative Comparison

Below we firstly present a few example of generated graphs for each model on each dataset they were trained on vis-a-vis the original training graphs. We also plot table for various structural metrics that were mentioned above. We then calculate and plot MMD values for each model on each dataset for comparison. We then discuss some interesting finds from these values.

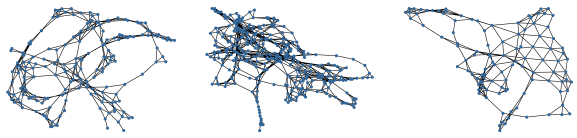

Figure 1: Training graphs from Enzymes data

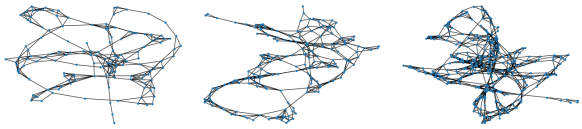

Figure 2: Enzyme graphs generated by NetGAN

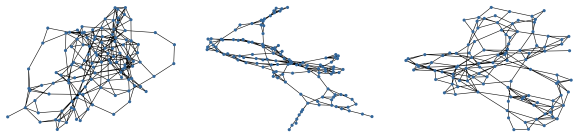

Figure 3: Enzyme graphs generated by GRAN

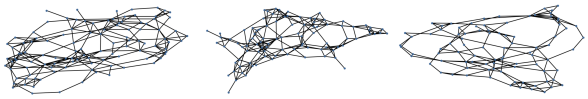

Figure 4: Enzyme graphs generated by GraphRNN

Discussions and Analysis

Synthetic Datasets

Grid.

(1) Qualitative :
- NetGans : While the generated graphs are not perfect grids, we can clearly see they are all planar graphs with lots of squares, few triangles, pentagons and hexagons.
- GRAN : The generated grid graphs look very decent, although twisted at times. Nevertheless, they sufficiently capture the uniform squares all over.

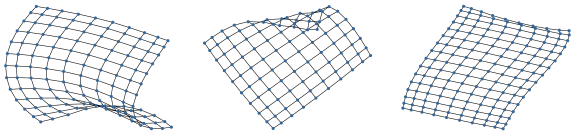

Figure 5: Training graphs from Grid data

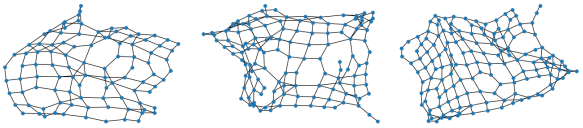

Figure 6: Grid graphs generated by NetGAN

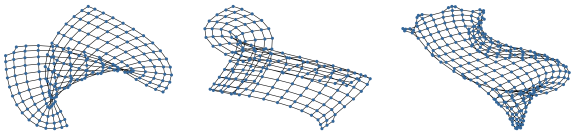

Figure 7: Grid graphs generated by GRAN

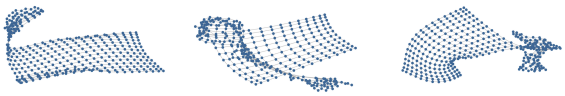

Figure 8: Grid graphs generated by GraphRNN

- GraphRNN : The generated graphs show an overall planar structure and conforms to the grid based shapes for most of the nodes. However, a set of nodes among those generated graphs forms a weird substructure where is goes onto form a nearly disconnected components isolated from the rest of the graph, except by a few edges. Visually, the planar structure is much better captured by the GraphRNN as compared to NetGAN.

(2) Quantitative :
- NetGAN performs poorly in terms of gini index of degrees on Grid dataset while the other two models have gini indices that are very close to the index of perfect grids. This can be observed easily from the generated graphs : the graphs from both GraphRNN and G-RNN have degrees typical of grid data (2, 3, 4), while generated graphs from NetGAN shows much larger variability. Nevertheless, the generated grid graphs still have interesting similarities visually.
- Wedge count (number of paths of length 2) of NetGANs is also relatively low for grid graphs as compared to GRAN and GraphRNN (which are super close to true values) which can be explained to some

Table 1: Comparison on Grid Dataset

Model	gini idx	num comp	claw ct.	min deg	wedge ct.	asrt.	LCC size	max deg	tri ct.	EdgDis entr.	avg deg	cpl	clust. coef.	pwrL exp
NetGAN	0.122	1.0	481.05	1.4	659.7	0.195	125.5	6.05	1.5	0.9927	3.636	6.427	0.009	2.190
GRAN	0.048	1.05	648.0	1.6	1047.6	0.588	202.1	4.0	0.7	0.9980	3.707	9.635	0.002	2.295
G-RNN	0.046	1.68	1277.6	1.8	1975.6	0.341	272.8	5.18	2.53	0.9983	3.802	15.83	0.006	2.423
Actual	0.044	1.0	883.0	2.0	1415.5	0.610	268.2	4.0	0.0	0.9984	3.748	11.0	0.0	2.616

Table 2: Comparison on Enzyme Dataset

Model	gini idx	num comp	claw ct.	min deg	wedge ct.	asrt.	LCC size	max deg	tri ct.	EdgDis entr.	avg deg	cpl	clust. coef.	pwrL exp
NetGAN	0.175	1.0	3828.2	1.15	2671.9	0.188	224.0	9.95	369.5	0.9891	4.951	6.434	0.301	1.721
GRAN	0.171	1.265	4694.4	1.66	3240.5	0.271	266.4	10.3	434.8	0.9901	5.005	6.930	0.411	2.006
G-RNN	0.184	1.15	5270.1	1.25	3132.4	0.098	230.4	10.6	252.7	0.9855	4.993	4.891	0.208	1.905
Actual	0.160	1.032	2795.3	1.39	2074.7	0.247	189.5	9.16	311.6	0.9901	4.875	7.546	0.358	1.868

Table 3: Comparison on Triangular Dataset

Model	gini idx	num comp	claw ct.	min deg	wedge ct.	asrt.	LCC size	max deg	tri ct.	EdgDis entr.	avg deg	cpl	clust. coef.	pwrL exp
NetGAN	0.183	1.0	2548.0	1.75	1841.8	0.162	145.5	9.15	207.4	0.9918	5.328	5.241	0.245	1.953
GRAN	0.26	1.1	307	1.0	351	-0.06	101	7.0	120	0.97	2.55	5.16	0.09	2.33
G-RNN	0.132	1.00	2967.3	1.25	2377.65	0.074	210.9	8.0	255.6	0.993	5.080	10.34	0.259	1.752
Actual	0.085	1.0	2432.2	2.0	1951.5	0.319	161.0	6.0	268.2	0.9948	5.304	7.510	0.331	2.058

Table 4: Comparison on Citeseer Dataset

Model	gini idx	num comp	claw ct.	min deg	wedge ct.	asrt.	LCC size	max deg	tri ct.	EdgDis entr.	avg deg	cpl	clust. coef.	pwrL exp
NetGAN	0.466	1.0	75112	1.0	5218.4	-0.265	173.7	59.3	179.2	0.8993	4.170	3.186	0.012	2.120
GRAN	0.414	1.0	74300	1.0	3360	-0.234	123	40.5	70	0.90	3.34	3.47	0.03	2.23
G-RNN	0.406	29.8	43573	1.0	4221.7	-0.022	183.8	42.1	80.25	0.940	3.308	3.975	0.021	2.457
Actual	0.435	1.0	78849	1.0	4811.4	-0.256	146.7	58.2	173.8	0.9011	4.188	3.147	0.022	2.090

Table 5: Comparison on Cora Dataset

Model	gini idx	num comp	claw ct.	min deg	wedge ct.	asrt.	LCC size	max deg	tri ct.	EdgDis entr.	avg deg	cpl	clust. coef.	pwrL exp
NetGAN	0.456	1.0	517247	1.0	12762	-0.248	276.2	124	196.4	0.8928	3.957	3.210	0.006	2.007
GRAN	0.422	1.1	127596	1.0	6433	-0.24	155.5	54	153	0.91	4.4	3.08	0.05	2.08
G-RNN	0.337	13.4	13948.8	1.0	2236	-0.076	180.9	34.8	60.9	0.959	2.736	7.619	0.029	2.276
Actual	0.440	1.0	215083	1.0	5502	-0.312	139.2	72.5	101.1	0.8851	3.492	3.156	0.020	2.226

extend by observing the generated graphs have larger polygons and smaller degrees in the center. Also the three graphs that NetGANs was trained on were relatively smaller graphs compared to the median

Table 6: Comparison on Hexagonal Dataset

Model	gini idx	num comp	claw ct.	min deg	wedge ct.	asrt.	LCC size	max deg	tri ct.	EdgDis entr.	avg deg	cpl	clust. coef.	pwrL exp
G-RNN	0.110	1.2	1031.35	1.0	2457.4	-0.005	826.25	5.1	18.4	0.995	2.890	51.137	0.053	1.975
Actual	0.040	1.0	266.4	2.0	847.8	0.212	315.00	3.0	0.00	0.998	2.843	13.695	0.00	3.925

Table 7: Comparison on Smaller grid Dataset

Model	gini idx	num comp	claw ct.	min deg	wedge ct.	asrt.	LCC size	max deg	tri ct.	EdgDis entr.	avg deg	cpl	clust. coef.	pwrL exp
G-RNN	0.158	2.7	499.7	1.0	1074.45	0.373	256.55	4.85	54.7	0.991	2.83	44.53	0.332	2.020
Actual	0.053	1.0	386.0	2.0	639.14	0.583	128.57	4.00	0.00	0.997	3.64	7.571	0.000	2.703

Table 8: Our MMD Metrics for Grid

Model	Degree	Clustering	Orbit	Spectral
NetGAN	0.126	0.0483	0.0027	0.0301
GRAN	0.5409	0.00395	0.008296	0.03203
GraphRNN	0.00274	0.02386	0.00722	0.03023

Table 9: Our MMD Metrics for Enzymes

Model	Degree	Clustering	Orbit	Spectral
NetGAN	0.002507	0.1118	0.135	0.0102
GRAN	0.0372	0.158	0.5879	0.00250
GraphRNN	0.0088	0.1436	0.44075	0.01933

Table 10: Our MMD Metrics for Triangular

Model	Degree	Clustering	Orbit	Spectral
NetGAN	0.16509	1.10902	0.4659	0.03559
GRAN	0.42	1.4	1.6	0.18
GraphRNN	0.079	1.150	0.279	0.043

Table 11: Our MMD Metrics for Citeseer

Model	Degree	Clustering	Orbit	Spectral
NetGAN	0.00294	0.125059	0.210839	0.011688
GRAN	0.0106	0.11	0.13	0.015
GraphRNN	0.009	0.389	0.231	0.100

Table 12: Our MMD Metrics for Cora

Model	Degree	Clustering	Orbit	Spectral
NetGAN	0.010765	0.204398	0.123714	0.037160
GRAN	0.016	0.41	0.0249	0.034
GraphRNN	0.002	0.695	0.281	0.133

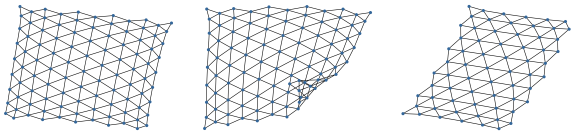

Figure 9: Training graphs for Triangle data

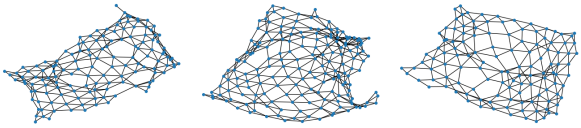

Figure 10: Triangle graphs generated by NetGAN

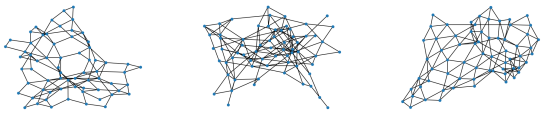

Figure 11: Triangle graphs generated by GRAN

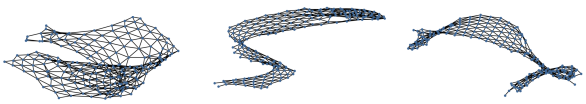

Figure 12: Triangle graphs generated by GraphRNN

grid graph size (evident from the small LCC size - NetGAN generally produce connected graphs) - so a count metric is not the best to assess NetGAN performance.

- The MMD metric for Spectra is very close to original values of the authors showing that the models do capture global properties well.

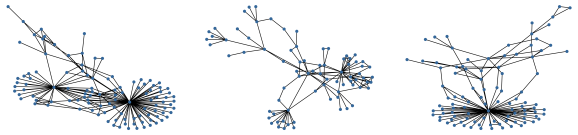

Figure 13: Training graphs Cora data

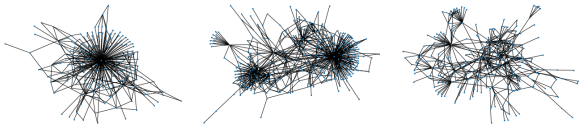

Figure 14: Cora graphs generated by NetGAN

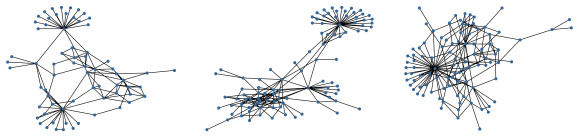

Figure 15: Cora graphs generated by GRAN

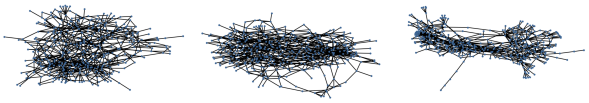

Figure 16: Cora graphs generated by GraphRNN

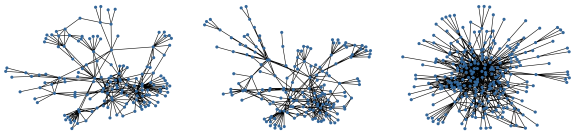

Figure 17: Training graphs on Citeseer data

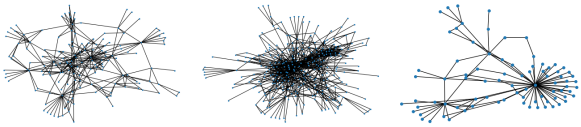

Figure 18: Citeseer graphs generated by NetGAN

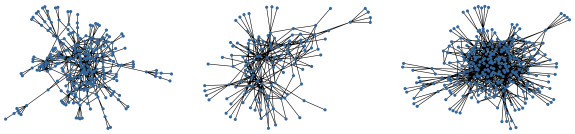

Figure 19: Citeseer graphs generated by GRAN

Figure 20: Citeseer graphs generated by GraphRNN

Triangular.

(1) Qualitative :
- NetGans : Broadly, the generated graphs show both planar and triangular structure, and one that is satisfied both locally and globally.
- GRAN : We can see that although the generated graphs are not truly planar, we have lot of induced triangle subgraphs showing that the model learnt the triangular structure.
- GraphRNN : We see that the generated graphs have triangular local structure. However, globally, the structure is not fully planar structure as seen by a set of nodes that are disconnected from rest, except by few edges.

(2) Quantitative :
- NetGans : The MMD values referring closely to the local structure in the generated graphs are comparable to the GraphRNN and lower than GRAN. The Spectral MMD, referring to the global structure is very close to zero, much better than both GRAN and GraphRNN as expected.
- GRAN : GRAN performs poorly in terms of metrics compared to both GraphRNN and NetGans for a triangular dataset (especially noticeable is the assortativity with negative sign). Both the local and global MMD values are quite high.
- GraphRNN : GraphRNN performs reasonably in terms of the data metrics in a new dataset, outperforming GRAN in all MMD statistics. The MMD statistics for degree is nearly 0 for the GraphRNN and this can be seen from the fact that the average degree is nearly the same as the average degree in true graphs.

Hexagonal.

(1) Qualitative :
- GraphRNN : We see that the generated graphs have triangular local structure. However, globally, the structure is not fully planar structure as seen by a set of nodes that are disconnected from rest, except by few edges.

(2) Quantitative :
- GraphRNN : GraphRNN performs reasonably in terms of the data metrics in a new dataset, outperforming GRAN in all MMD statistics. The MMD statistics for degree is nearly 0 for the GraphRNN and this can be seen from the fact that the average degree is nearly the same as the average degree in true graphs.

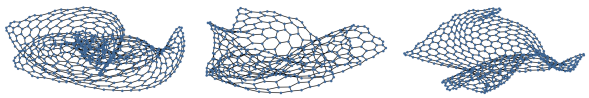

Figure 21: Training graphs on Hexagonal data

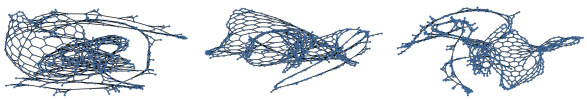

Figure 22: Hexagonal graphs generated by GraphRNN

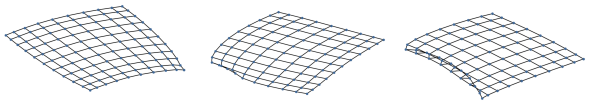

Figure 23: Training graphs on Smaller Grid data

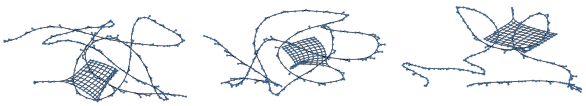

Figure 24: Smaller Grid graphs generated by GraphRNN

Smaller Grid.

(1) Qualitative :
- GraphRNN : Similar to the grid graphs, graphs for model trained on smaller grids shows overall planar structure. However, a set of nodes forms weird substructure of linear parallel chains, disconnected from rest, except by few edges. This effect is pronounced in the smaller grids as compared to the grid datasets.

(2) Quantitative :
- GraphRNN : The average degree is quite different for the generated graphs as compared to the actual graphs, likely due to the linear chains of nodes. There is a high clustering coefficients in the generated graphs, which suggests the local structure is not replicated for the smaller grid graphs.

Real-World Datasets

Enzymes.

(1) Qualitative :
- NetGans : The generated enzyme graphs look reasonable with many nodes in large rings and a few long chain typical of most enzymes.
- GRAN : For the enzyme dataset we observe that the graphs tend to have a much lower number of nodes than the training graphs although they do capture the overall structure well. We also observe that we are able to capture rings in the protein structure even though we are most probably underfitting due to the low number of iterations we ran the model for.
- GraphRNN : For the enzymes dataset, the circular and the triangular shapes, which are observed in the training graphs are captured by the GraphRNN models. Though, it does seem that the model is underfitting the original data and capturing a specific set of substructures in the generation.

(2) Quantitative :
- The GRAN model tends to generate too many triangles resulting in a higher clustering coefficient than correct. We can further confirm this with the poor MMD Clustering Coefficient value it obtains.
- The MMD spectral value for all three models is very close to what the authors proposed. This leads us to observe that the GraphRNN and GRAN models tend to learn the global structure of the graphs in the initial iterations(as the enzyme dataset was trained for a small number of iterations). Similarly, the select training graphs chosen for NetGAN represent the distribution well.

Table 13: Our MMD Metrics for Smaller grid Dataset

Model	Degree	Clustering	Orbit	Spectral
GraphRNN	0.203	1.804	0.482	0.149

Table 14: Our MMD Metrics for Hexagonal

Model	Degree	Clustering	Orbit	Spectral
GraphRNN	0.031	0.343	0.005	0.032

Table 15: MMD Metrics comparison with paper(Grid)

Model	Degree	Clustering	Orbit	Spectral
GRAN(Orig)	0.000822	0.00379	0.001587	0.01603
GRAN(Ours)	0.5409	0.00395	0.008296	0.03203
G-RNN(Orig)	$10^{-}5$	0	$10^{-}5$	N-A
G-RNN(Ours)	0.00274	0.02386	0.00722	0.03023

Table 16: MMD Metrics comparison with paper(Enzyme)

Model	Degree	Clustering	Orbit	Spectral
GRAN(Orig)	0.00198	0.0486	0.1306	0.0051
GRAN(Ours)	0.0372	0.158	0.5879	0.00250
G-RNN(Orig)	0.034	0.935	0.217	N-A
G-RNN(Ours)	0.0088	0.1436	0.44075	0.01933

Citeseer.

(1) Qualitative :
- NetGans : The generated graphs for Cora produce graphs few very influential nodes which is expected from the way we sampled our subgraphs (using 3-hop neighbourhoods of a random graph)
- GRAN : We can observe that the model captures the ego network structure well. It replicates the hub (multiple nodes coming out of 1 node) structures which are ubiquitous in Citeseer well.
- GraphRNN : The most striking observation is that the generated graphs are often disconnected with several small components, but otherwise okay.

(2) Quantitative :
- NetGans : On the whole, NetGAN is closest to the actual statistics for most metrics in Citeseer data suggesting that the 3 training graphs used represent the underlying graph distribution pretty well.
- GRAN : GRAN does reasonably well on most metrics, but seem to be producing relatively smaller graphs. This in turn explains the low wedge count and triangle counts.
- GraphRNN : GraphRNN tends to produce several small disconnected components in general, which in turn seem to affect other statistics like claw count and triangle count. This might be because of the presence of a lot of low degree nodes in the training data (because of the nature of our sampling : induced 3-hop subgraphs).

Cora.

(1) Qualitative :
- NetGans : Looks like all 3 sub-graphs chosen to train NetGAN are denser than typical 3-hop subgraphs. Nevertheless, we can see the prominent star pattern (few high degree nodes) again.
- GRAN : The model overall produces nets similar to training graphs. The most striking feature for these real world ego networks is the subgraph in which 1 node has links to many other nodes (hubs) which is being captured very well.
- GraphRNN : The images shown are some of the best images from GraphRNN. Just like Citeseer data, generated graphs are very disconnected in Cora dataset also.

(2) Quantitative :
- NetGans : Once again, NetGAN performs reasonably well on most metrics. Size of LCC, triangle count, claw count and wedge count are a bit higher because the chosen training graphs were larger than the mean size. Nevertheless, metrics that look at average properties are well captured.
- GRAN : GRAN has similar values for most metrics compared to actual. The average node degree is slightly higher, which probably explains slightly low claw count and slightly high triangle count.
- GraphRNN : Once again, GraphRNN produces graphs with several components in general, giving rise to lower claw count, triangle count and max degree

7 CONCLUSIONS AND TAKE-AWAYS

- The training time for NetGANs for each individual model learnt was considerable lower than the other two models and gave nearly similar results! This is expected since NetGANs exploits the local structure of the graphs using random walks, effectively running in almost linear time.
- The GRAN models clearly performs much better on real world datasets as compared to synthetic datasets. Thus we can conclude that the model is not very good at learning regular patters but can capture substructures like hubs, claws, stars very well.
- We also see GraphRNN faltering on synthetic datasets. Thus for synthetic datasets, it might be a better option to go for NetGans.
- However since NetGans trains on only 1 graph, it might not be suitable for real-world datasets as there can be a lot of variation across training graphs which GRAN, GraphRNN can capture but NetGan must be trained separately on these multiple training graphs for the same effect.
- Although MMD values of these models are not excitingly good, the generated graphs are visually much more acceptable. This implies that this measure might not have as high a correlation with the visual acceptance as we have been believing.
- We connected with the authors of these papers, and it seems that the reported results for the models were highly optimized for publication.
- Re-training these models even on the same datasets is a challenging feat and the entire DL community needs to find ways to improve the stability of these models.

8 FUTURE STEPS

- As mentioned the MMD might not be the best metric, so an obvious extension is to calculate the FID scores for the generated graphs. However, the issue is that we need a parallel for the Inception network for our case and no known pre-trained model trained on a huge graph classification dataset exists. Thus the first step for this would be to train a huge graph classification network on same standard large classification dataset and then use it for FID calculation.

REFERENCES

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875* (2017).

[2] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. 2018. Netgan: Generating graphs via random walks. *arXiv preprint arXiv:1803.00816* (2018).

[3] Paul D Dobson and Andrew J Doig. 2003. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology* 330, 4 (2003), 771–783.

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

[5] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. 1998. CiteSeer: An Automatic Citation Indexing System.. In *ACM DL*. 89–98.

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

[7] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[8] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[9] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[10] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. 2010. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research* 11, Feb (2010), 985–1042.

[11] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Charlie Nash, William L Hamilton, David Duvenaud, Raquel Urtasun, and Richard S Zemel. 2019. Efficient Graph Generation with Graph Recurrent Attention Networks. *arXiv preprint arXiv:1910.00760* (2019).

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

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

[14] Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. *arXiv preprint arXiv:1802.08773* (2018).