GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models

Jiaxuan You*[1] Rex Ying*[1] Xiang Ren[2] William L. Hamilton[1] Jure Leskovec[1]

1: Stanford University, 2: University of Southern California, LA

ICML 2018

Presenter: Arshdeep Sekhon

https://qdata.github.io/deep2Read

Motivation

- Most previous generative models use a priori structural assumptions: degree distribution, community structure, etc.
- But we want to learn directly from observed set of graphs.
- Deep generative models that learn from data: VAE, GAN,etc.

Challenges for Graph Generation

- **Large Output Space**: For an n nodes graph specify n^2 values for output structure

Challenges for Graph Generation

- **Large Output Space**: For an n nodes graph specify n^2 values for output structure
- **Large Variable Space**: variable n nodes and e edges

Challenges for Graph Generation

- **Large Output Space**: For an n nodes graph specify n^2 values for output structure
- **Large Variable Space**: variable n nodes and e edges
- **Non-unique representations**: Same graph$\rightarrow$ $n!$ adjacency matrices $\rightarrow$ $n!$ node orderings
  - Optimize objective ? Reconstruction error?

Challenges for Graph Generation

- **Large Output Space**: For an n nodes graph specify n^2 values for output structure
- **Large Variable Space**: variable n nodes and e edges
- **Non-unique representations**: Same graph$\rightarrow$ $n!$ adjacency matrices $\rightarrow$ $n!$ node orderings
  - Optimize objective ? Reconstruction error?
- **Complex Dependencies**: edges not independent

GraphRNN: Method Overview

- **Task**: learn $p_{model}(G)$ based on observed training set of graphs $\{G_1, \ldots, G_s\}$ from true $p(G)$
- each graph G_i has variable nodes and edges

GraphRNN: Method Overview

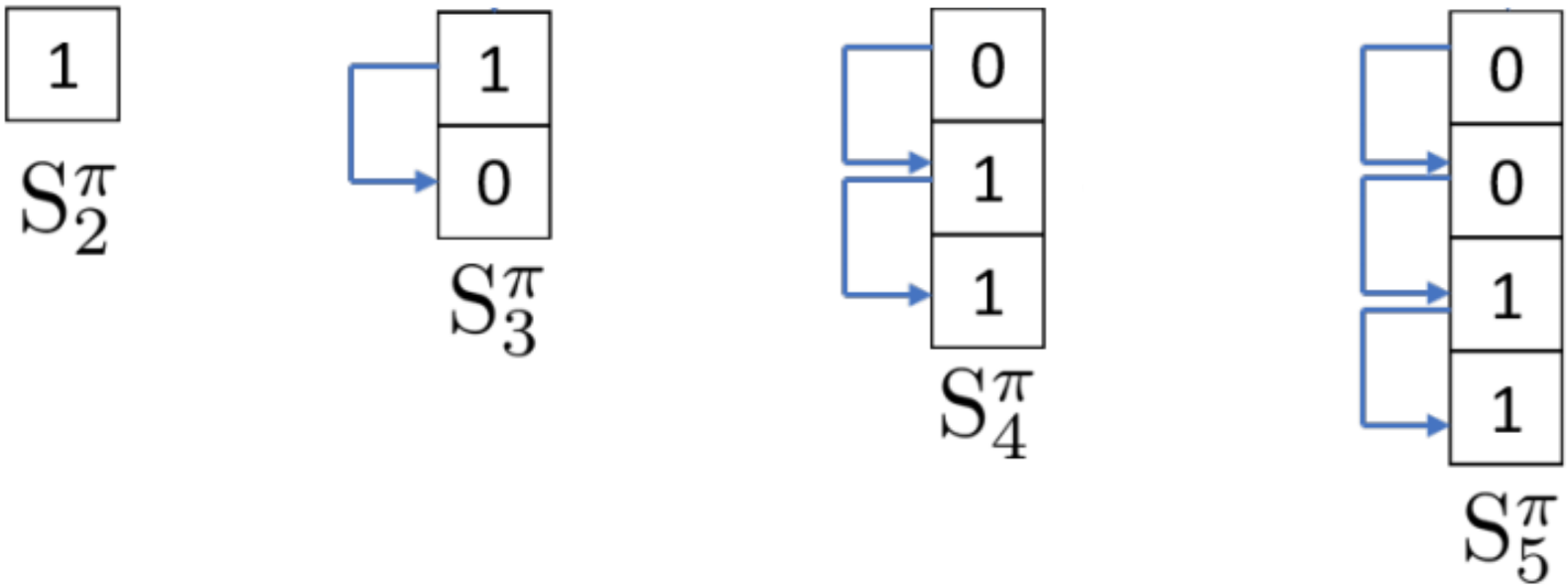
- **Task**: learn $p_{model}(G)$ based on observed training set of graphs $\{G_1, \ldots, G_s\}$ from true $p(G)$
- each graph G_i has variable nodes and edges
- **Method**: Autoregressive generation – a sequence of additions of new nodes and edges
- trained using stochastic gradient descent with a maximum likelihood loss

Step 1: Model Graphs as Sequences

- define a mapping f_S from graphs to sequences
- for a graph $G \sim p(G)$ with n nodes under node ordering π

$$S^\pi = f_S(G, \pi) = (S_1^\pi, ..., S_n^\pi), \tag{1}$$

$$S_i^\pi = (A_{1,i}^\pi, ..., A_{i-1,i}^\pi)^T, \forall i \in \{2, ..., n\}. \tag{2}$$



Graphs as Sequences

- Instead of learning $p(G)$: learn $p(S^{\pi})$

Graphs as Sequences

- Instead of learning $p(G)$: learn $p(S^{\pi})$

$$p(G) = \sum_{S^{\pi}} p(G|S^{\pi}) \tag{3}$$

$$p(G) = \sum_{S^{\pi}} p(S^{\pi})\ \mathbb{1}[f_G(S^{\pi}) = G], \tag{4}$$

Graphs as Sequences

- Instead of learning $p(G)$: learn $p(S^\pi)$

$$p(G) = \sum_{S^\pi} p(G|S^\pi) \tag{3}$$

$$p(G) = \sum_{S^\pi} p(S^\pi)\ \mathbb{1}[f_G(S^\pi) = G], \tag{4}$$

- sequential nature of S^π: Sample from $p(S^\pi)$, can be modeled autoregressively

$$p(S^\pi) = \prod_{i=1}^{n+1} p(S_i^\pi | S_1^\pi, ..., S_{i-1}^\pi) \tag{5}$$

Step 2: Generation Framework (Graph Level RNN)

- $p(S_i^\pi | S_{<i}^\pi)$: capture how current node connected to previous nodes based on how the previous nodes are connected to each other
- $p(S_i^\pi | S_{<i}^\pi)$:using NNs to paramterize

$$h_i = f_{\text{trans}}(h_{i-1}, S_{i-1}^\pi), \tag{6}$$
$$\theta_i = f_{\text{out}}(h_i), \tag{7}$$

- θ_i specifies the distribution of next node's adjacency vector ($S_i^\pi \sim \mathcal{P}_{\theta_i}$).
- $\mathcal{P}_{\theta_i}$ can be an arbitrary distribution over binary vectors.

Algorithm 1 GraphRNN inference algorithm

Input: RNN-based transition module f_{trans}, output module f_{out}, probability distribution $\mathcal{P}_{\theta_i}$ parameterized by θ_i, start token SOS, end token EOS, empty graph state h'

Output: Graph sequence S^π

$S_1^\pi = \text{SOS}, h_1 = h', i = 1$

repeat

$i = i + 1$

$h_i = f_{\text{trans}}(h_{i-1}, S_{i-1}^\pi)$ {update graph state}

$\theta_i = f_{\text{out}}(h_i)$

$S_i^\pi \sim \mathcal{P}_{\theta_i}$ {sample node i's edge connections}

until S_i^π is EOS

Return $S^\pi = (S_1^\pi, ..., S_i^\pi)$

Step 3: Edge Level RNN

- To fully capture complex edge dependencies,decompose $p(S_i^\pi | S_{<i}^\pi)$:

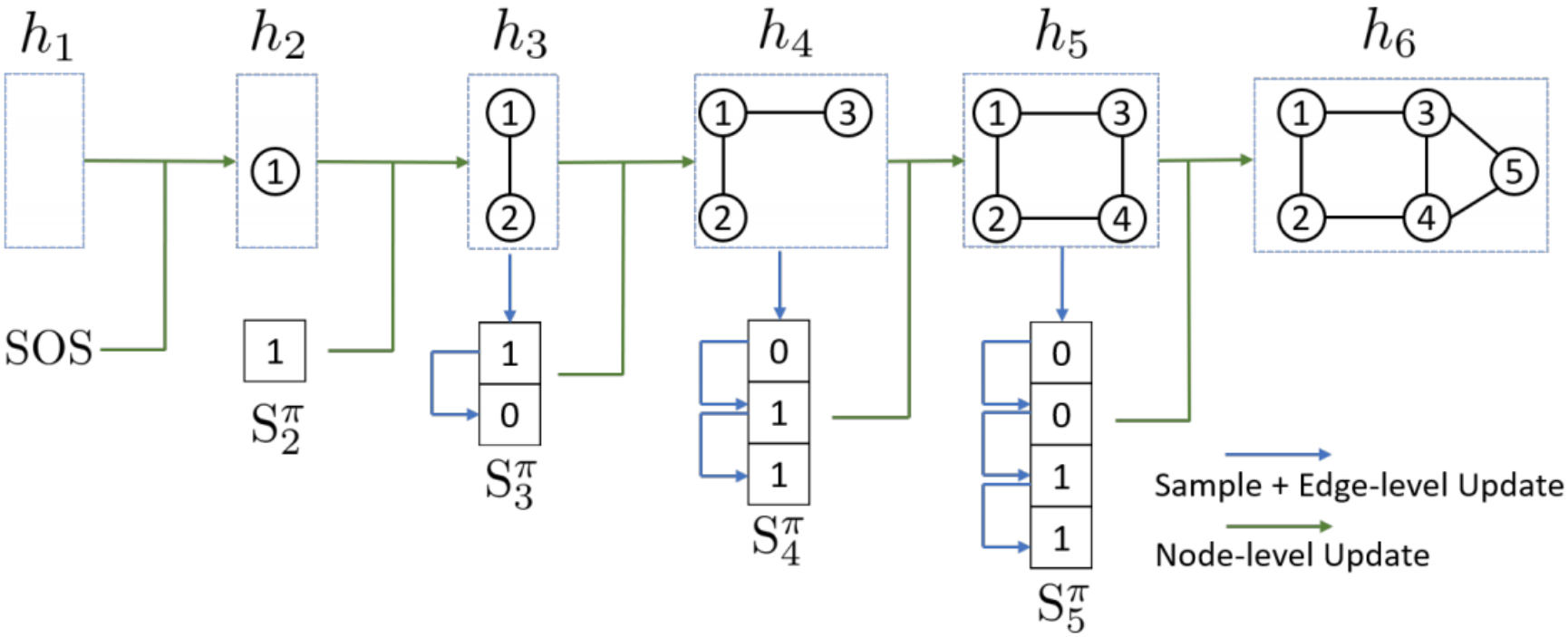
$$p(S_i^\pi | S_{<i}^\pi) = \prod_{j=1}^{i-1} p(S_{i,j}^\pi | S_{i,<j}^\pi, S_{<i}^\pi), \tag{8}$$

- each distribution in the product is approximated by an RNN

Step 3: Edge Level RNN

A Hierarchical RNN:

- Graph Level RNN: where the first RNN maintains the state of the graph
- Edge Level RNN: second RNN generates the edges of a given node.



Tractability: Breadth First Search

- any possible node permutation $\rightarrow$ generate graphs using breadth-first-search (BFS) node orderings

Change

$$S^{\pi} = f_S(G, \pi) = (S_1^{\pi}, ..., S_n^{\pi}), \tag{9}$$

To

$$S^{\pi} = f_S(G, \mathrm{BFS}(G, \pi)), \tag{10}$$

Tractability: Breadth First Search

Advantages:

- only need to train on all possible BFS orderings
- makes learning easier by reducing the number of edge predictions we need to make in the edgelevel RNN

Evaluation of Generated Graphs

- requires a comparison between two sets of graphs
- based on MMD score

$$MMD^2(p||q) = \mathbb{E}_{x,y\sim p}[k(x,y)] + \mathbb{E}_{x,y\sim q}[k(x,y)] - 2\mathbb{E}_{x\sim p,y\sim q}[k(x,y)]. \tag{11}$$

- graph distance hard to calculate: use graph statistics for distance measuring
- Use Wasserstein distance as distance metric calculator:

$$W(p,q) = \inf_{\gamma\in\Pi(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[||x-y||], \tag{12}$$

- Proposition: The kernel function defined by $k_W(p,q) = \exp\frac{W(p,q)}{2\sigma^2}$ induces a unique RKHS.

Datasets

- **Community** 500 two-community graphs with $60 \leq |V| \leq 160$. Each community is generated by the (E-R) Model with $n = |V|/2$ nodes and $p = 0.3$. We then add $0.05|V|$ inter-community edges with uniform probability.
- **Grid** 100 standard 2D grid graphs with $100 \leq |V| \leq 400$.
- **B-A** 500 graphs with $100 \leq |V| \leq 200$ that are generated using the BA model.
- **Protein** 918 protein graphs with $100 \leq |V| \leq 500$. Each protein is represented by a graph, where nodes are amino acids and two nodes are connected if they are less than 6 Angstroms apart.
- **Ego** 757 3-hop ego networks extracted from the Citeseer network with $50 \leq |V| \leq 399$

Quantitative Results: Evaluation with graph statistics

Table 1. Comparison of GraphRNN to traditional graph generative models using MMD. $(\max(|V|), \max(|E|))$ of each dataset is shown.

	Community (160,1945)			Ego (399,1071)			Grid (361,684)			Protein (500,1575)		
	Deg.	Clus.	Orbit	Deg.	Clus.	Orbit	Deg.	Clus.	Orbit	Deg.	Clus.	Orbit
E-R	0.021	1.243	0.049	0.508	1.288	0.232	1.011	0.018	0.900	0.145	1.779	1.135
B-A	0.268	0.322	0.047	0.275	0.973	0.095	1.860	0	0.720	1.401	1.706	0.920
Kronecker	0.259	1.685	0.069	0.108	0.975	0.052	1.074	0.008	0.080	0.084	0.441	0.288
MMSB	0.166	1.59	0.054	0.304	0.245	0.048	1.881	0.131	1.239	0.236	0.495	0.775
GraphRNN-S	0.055	0.016	0.041	0.090	**0.006**	0.043	0.029	10^{-5}	0.011	0.057	**0.102**	**0.037**
GraphRNN	**0.014**	**0.002**	**0.039**	**0.077**	0.316	**0.030**	$\mathbf{10^{-5}}$	**0**	$\mathbf{10^{-4}}$	**0.034**	0.935	0.217

Figure: . Comparison of GraphRNN to traditional graph generative models using MMD.

Quantitative Results: Evaluation with graph statistics

	Community-small (20,83)					Ego-small (18,69)				
	Degree	Clustering	Orbit	Train NLL	Test NLL	Degree	Clustering	Orbit	Train NLL	Test NLL
GraphVAE	0.35	0.98	0.54	13.55	25.48	0.13	0.17	0.05	12.45	14.28
DeepGMG	0.22	0.95	0.40	106.09	112.19	0.04	0.10	0.02	21.17	22.40
GraphRNN-S	**0.02**	0.15	**0.01**	31.24	35.94	0.002	**0.05**	**0.0009**	8.51	9.88
GraphRNN	0.03	**0.03**	**0.01**	28.95	35.10	**0.0003**	**0.05**	**0.0009**	9.05	10.61

Figure: . Comparison of GraphRNN to traditional graph generative models using MMD.

Quantitative Results: Graph Statistics

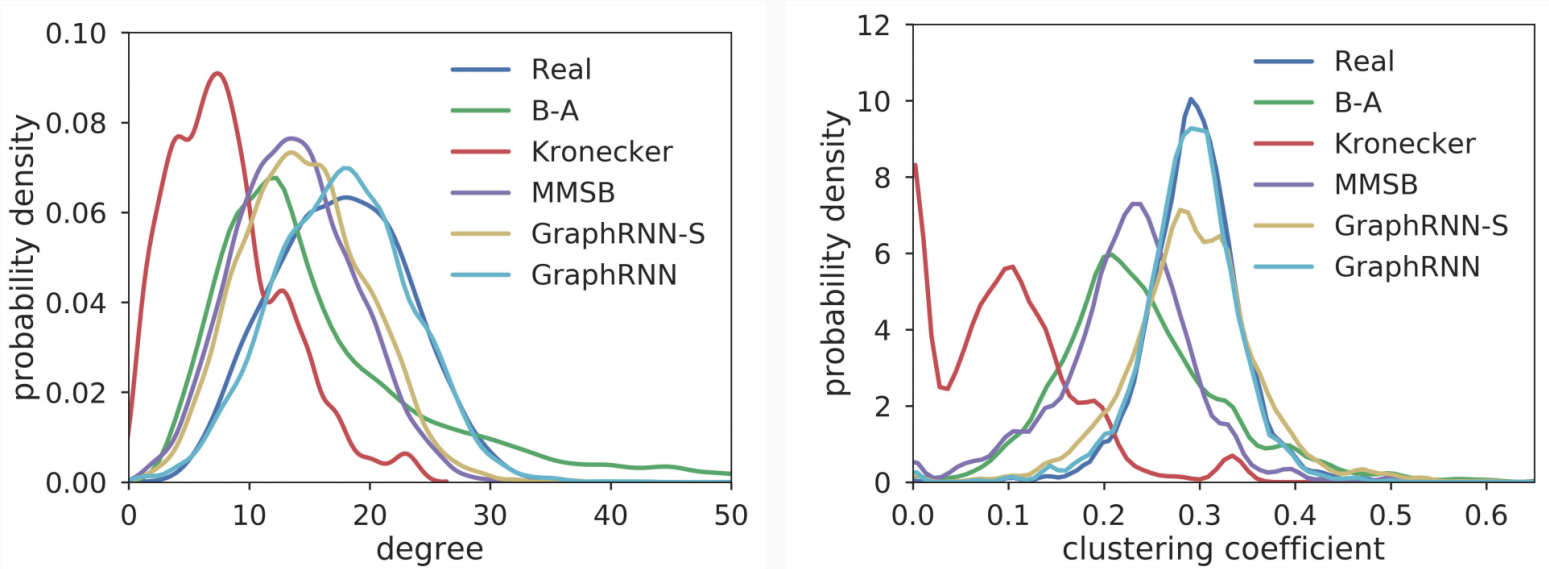


Figure 3: Average degree (Left) and clustering coefficient (Right) distributions of graphs from test set and graphs generated by GraphRNN and baseline models.

Quantitative Results: Robustness

Interpolate between (B-A) and (E-R) graphs

Randomly perturb $[0\%, 20\%, ..., 100\%]$ edges of B-A graphs

0% (B-A) $\longleftrightarrow$ 100% (E-R)

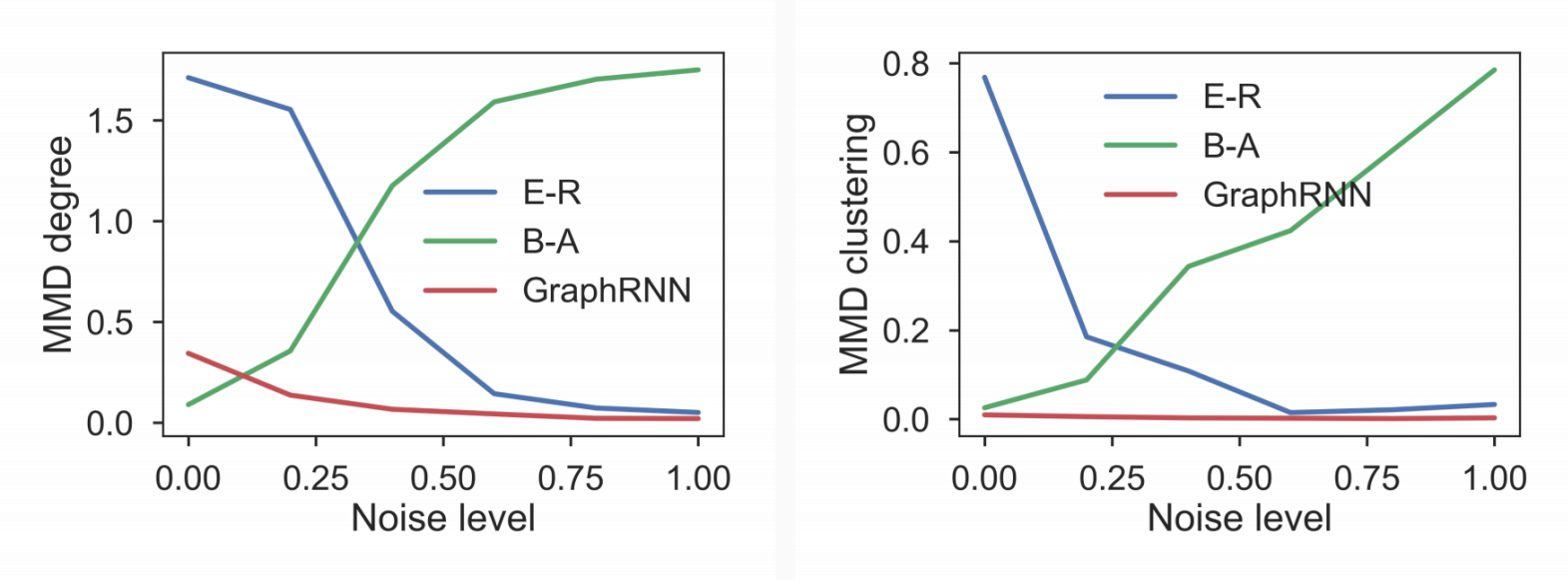


Figure 4: MMD performance of different approaches on degree (Left) and clustering coefficient (Right) under different noise level.

Graph Visualization

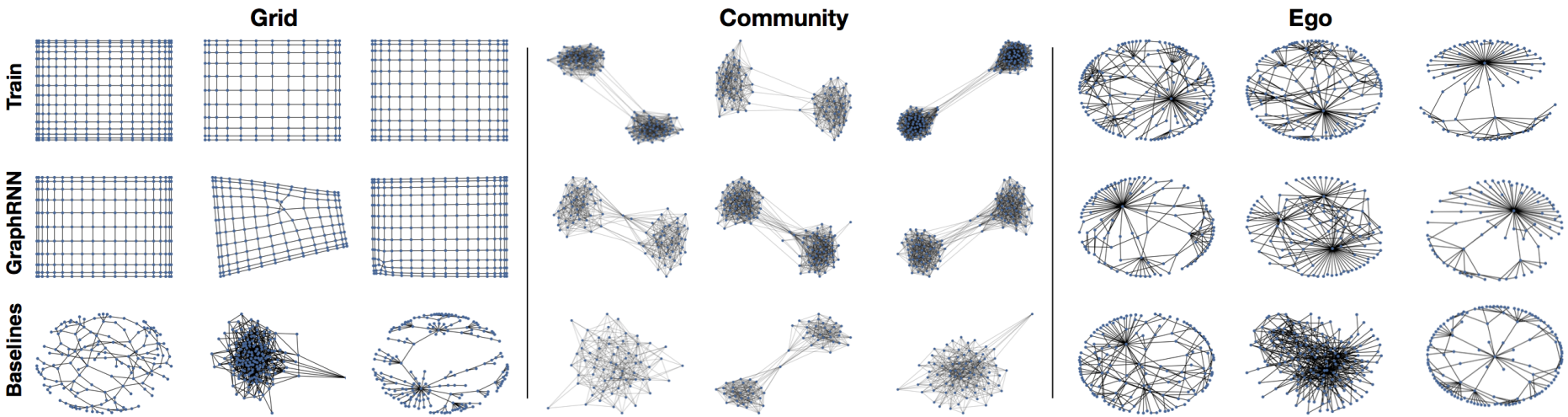


Figure: Visualization of graphs from grid dataset (Left group), community dataset (Middle group) and Ego dataset (Right group).

Conclusion

- Autoregressive way to generate graph sequences
- can generate variable sized graphs
- generates the adjacency matrix of a graph by generating the adjacency vector of each node step by step