C-GIN Network Completion

Completing Networks by Learning Local Connection Patterns

Zhang Zhang[ab], Ruyi Tao.[ab†], Yongzai Tao[c†], Mingze Qi[d], Jiang Zhang[ab*]

[a]School of Systems Science, Beijing Normal University

[b]Swarma Research, Beijing, China

[c]College of Computer Science and Technology, Zhejiang University

[d]College of Science, National University of Defense Technology

[†]Those author contribute equally

[*]Corresponding author: Jiang Zhang; zhangjiang@bnu.edu.cn

Abstract

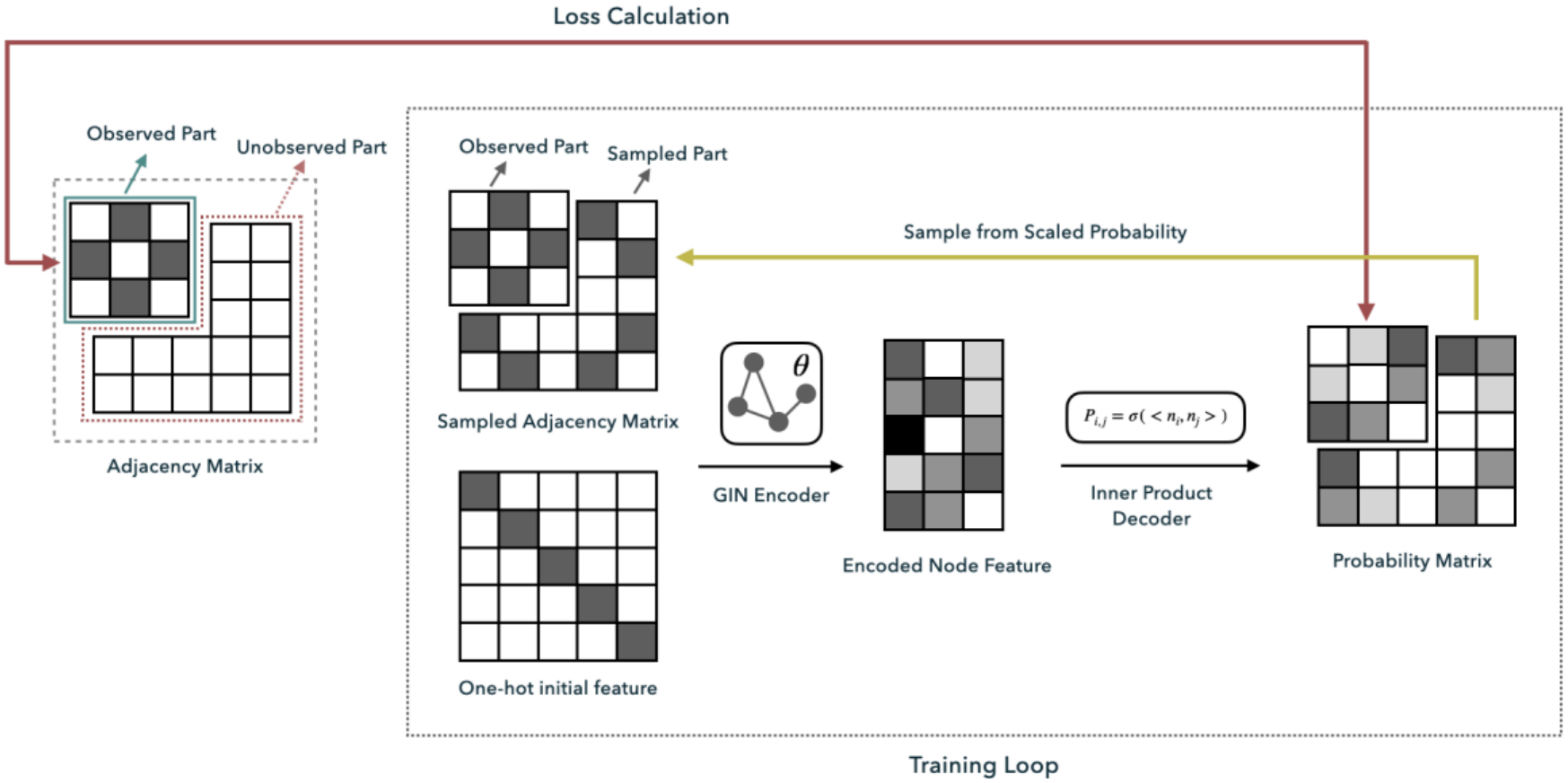
Network completion is a harder problem than link prediction because it does not only try to infer missing links but also nodes. Different methods have been proposed to solve this problem, but few of them employed structural information - the similarity of local connection patterns. In this paper, we propose a model named C-GIN to capture the local structural patterns from the observed part of a network based on the Graph Auto-Encoder framework equipped with Graph Isomorphism Network model and generalize these patterns to complete the whole graph. Experiments and analysis on synthetic and real-world networks from different domains show that competitive performance can be achieved by C-GIN with less information being needed, and higher accuracy compared with baseline prediction models in most cases can be obtained. We further proposed a metric "Reachable Clustering Coefficient(CC)" based on network structure. And experiments show that our model perform better on a network with higher Reachable CC.

张章

陶如意

https://arxiv.org/pdf/2204.11852.pdf

C-GIN Network Completion



Encoder:

$$H = GIN_\theta(\hat{A}, X),$$

Decoder:

$$P_{i,j} = \frac{1}{1 + \exp(-\langle H^i, H^j \rangle)}$$

Loss function:

$$\mathcal{L}(\theta) = -\sum_{i=0}^{N_O-2} \sum_{j=i+1}^{N_O} [A_{ij} \log P_{ij} + (1 - A_{ij}) \log(1 - P_{ij})]$$

Scale factor:

$$\gamma = \frac{N^2 - N_O^2}{N_O^2} \cdot \frac{\sum_{i,j<N_O} A_{ij}}{\sum_{i,j<N} P_{ij} - \sum_{i,j<N_O} P_{ij}}$$

https://arxiv.org/pdf/2204.11852.pdf

Results

Networks	PA	Random-De	KronEM	G-GCN	C-GIN
BA	58.67 ±1.5	59.33 ±1.5	64.1 ±0.6	74.67 ±2.1	**76.49 ±1.8**
WS	35.45 ±0.5	35.91 ±0.4	80.42 ±1.2	81.20 ±0.6	**85.22 ±0.3**
Kron	71.60 ±0.6	71.13 ±1.1	**83.9 ±0.4**	68.38 ±1.1	71.55 ±1.4
FF	79.33 ±0.9	74.16 ±0.7	62.11 ±0.4	**82.75 ±0.7**	80.17±1.8

Table 1: **AUC on unobserved part of synthetic networks:** In this table we show the experimental results of different methods on the synthetic networks.

https://arxiv.org/pdf/2204.11852.pdf

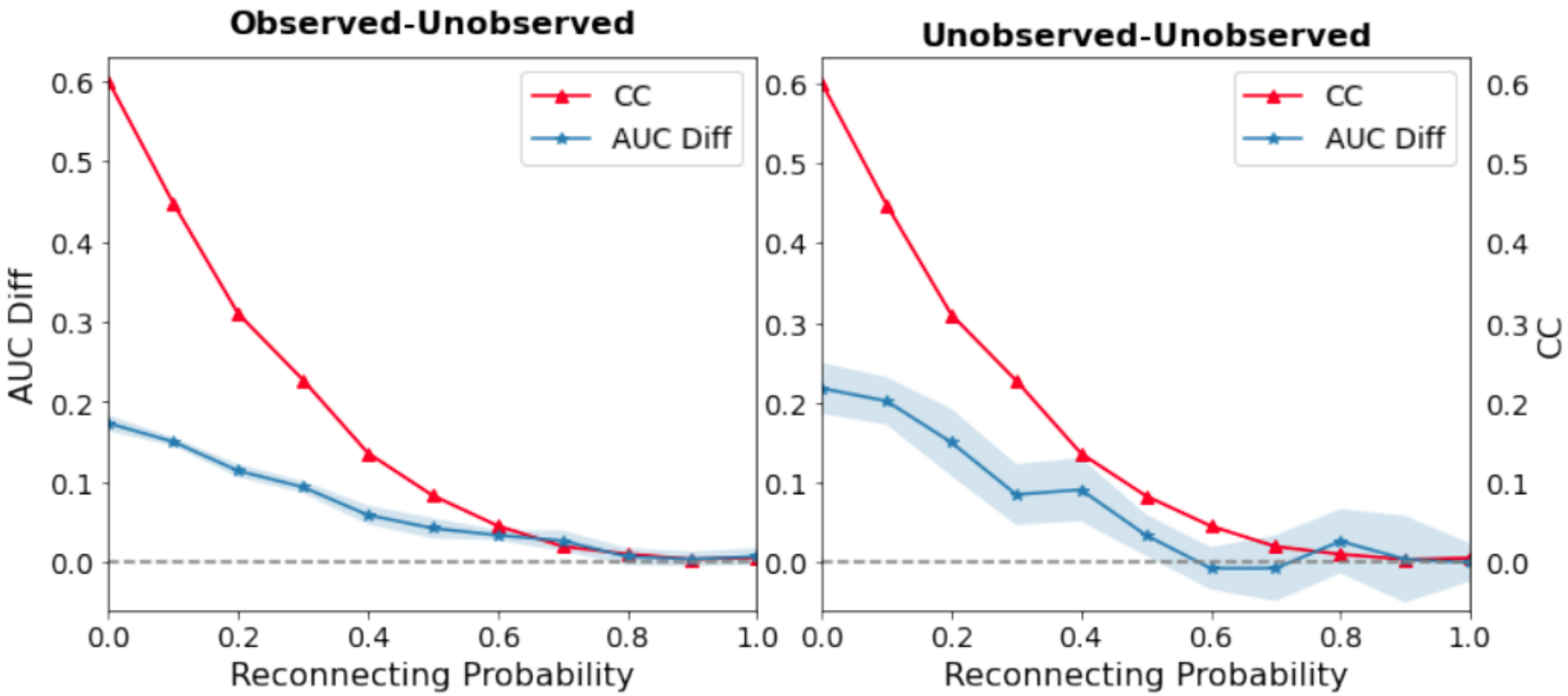
Results

Networks	AUC	Models			
		PA	G-GCN	C-GIN	Random-De
Bio_S	All	72.89 ± 1.3	83.23±1.8	**88.71±2.1**	70.22±0.7
	Observed-Unobserved	73.5 ± 1.4	88.83±1.7	**90.16±1.7**	72.41±0.9
	Unobserved-Unobserved	-	57.19±3.3	**74.05±3.4**	54.34±2.2
Bio_D	All	75.18 ±0.9	81.35±1.0	**85.79±4.0**	62.43±1.6
	Observed-Unobserved	77.01 ± 0.8	85.86±1.0	**88.48±2.6**	63.99±1.8
	Unobserved-Unobserved	-	57.07±1.1	**66.71±11.2**	54.66±2.7
Cora	All	60.01 ± 0.9	86.02±0.9	**87.37±0.3**	78.28±1.4
	Observed-Unobserved	60.13 ± 0.6	**92.07±0.9**	89.10±0.8	81.28±1.1
	Unobserved-Unobserved	-	55.58±2.3	**74.70±2.2**	53.78±2.1
Co-Author	All	58.88 ± 2.7	85.82±1.7	**91.61±1.6**	73.93±1.5
	Observed Uunobserved	59.07 ± 2.2	97.17±1.1	**93.78±1.8**	76.27±1.6
	Unobserved-Unobserved	-	57.36±7.0	**75.29±7.6**	60.66±4.1

Table 3: **AUC on unobserved part of synthetic networks:** In this table we show the experiment result of different methods on synthetic networks.

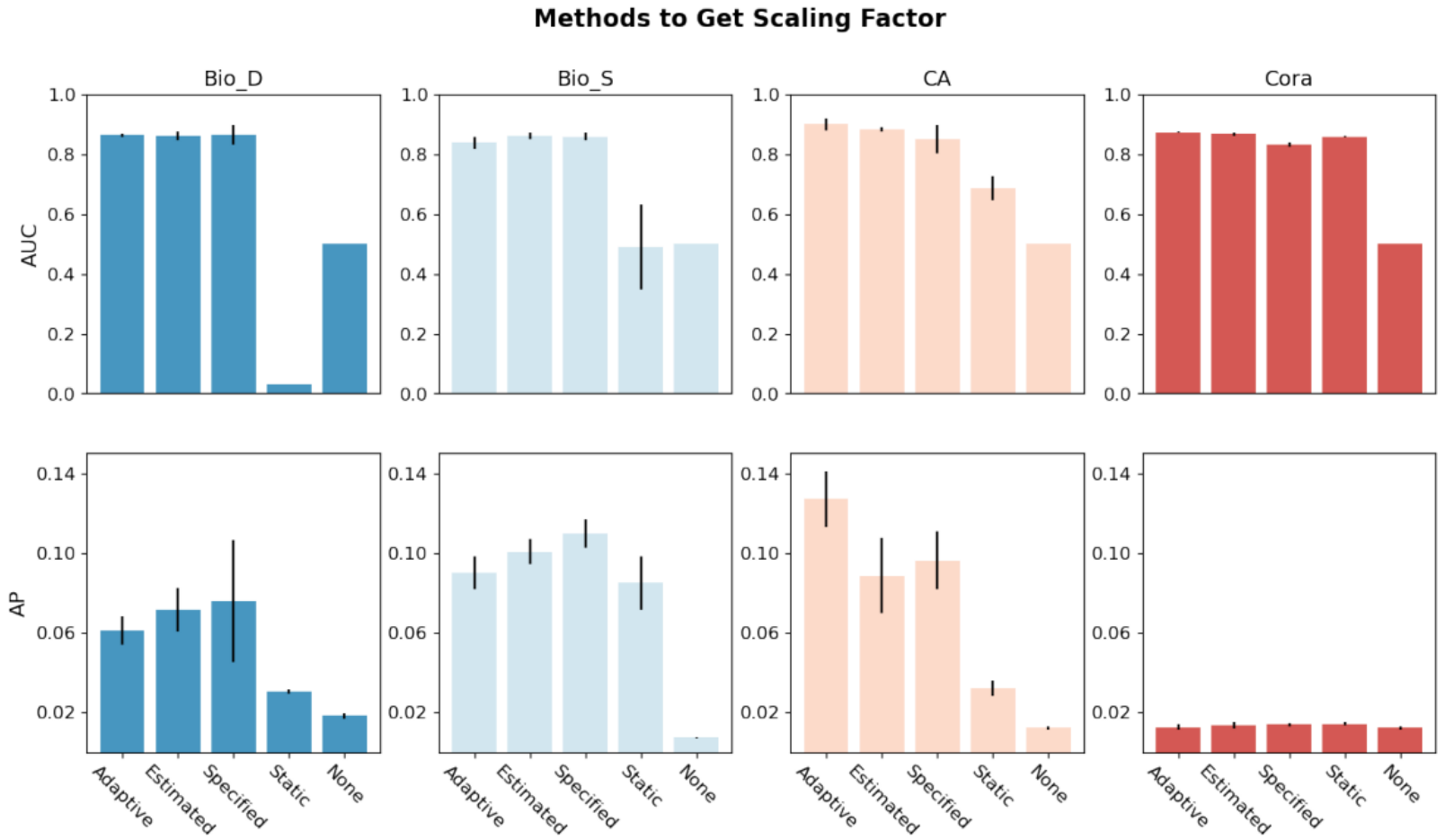
https://arxiv.org/pdf/2204.11852.pdf

Results



https://arxiv.org/pdf/2204.11852.pdf

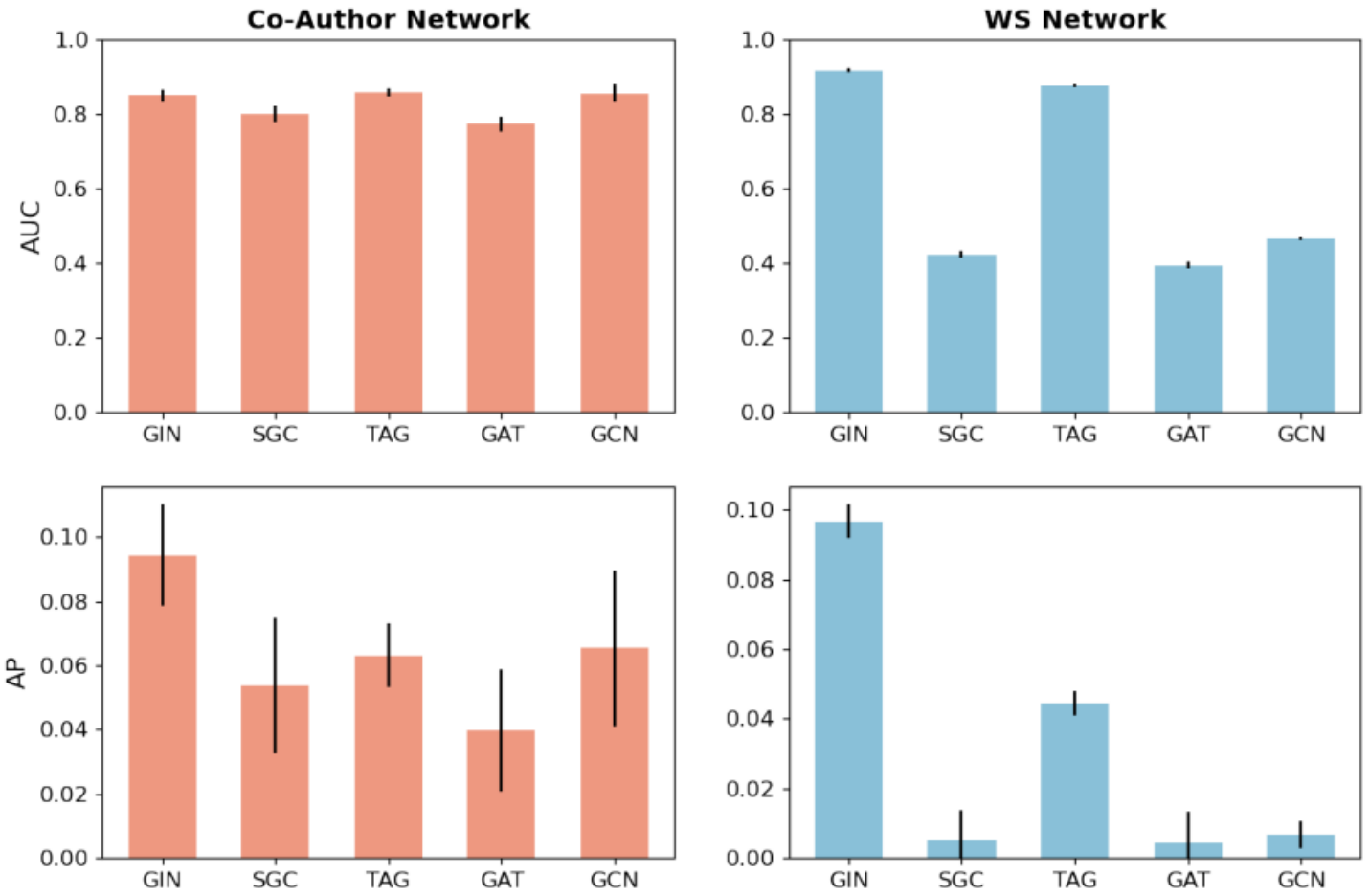
Results



- None: We set scaling factor as 1, which means there is no scaling operation.
- Specified: We set scaling factor as a specific value(0.001 here).
- Static: In this way, we use the decoded probability matrix at the first time in the training process, and calculate the scaling factor by Eq.6
- Estimated: In this way, we replace the decoded probability matrix with a uniform random matrix, and then use Eq.6 to calculate the scaling factor.

https://arxiv.org/pdf/2204.11852.pdf

Results



- Graph Convolutional Network(GCN)
- Graph Attention Network(GAT)
- Topology Adaptive Graph Convolutional Network(TAG)
- Simplifying Graph Convolutional Networks(SGC)

https://arxiv.org/pdf/2204.11852.pdf

Graph Isomorphism Network

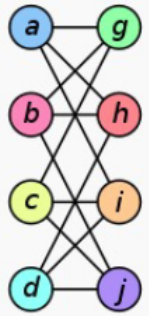
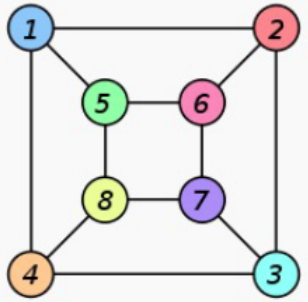
Published as a conference paper at ICLR 2019

How Powerful are Graph Neural Networks?

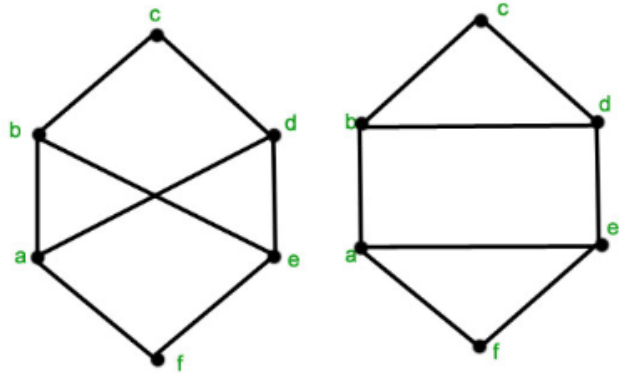
Keyulu Xu *† MIT keyulu@mit.edu
Weihua Hu *‡ Stanford University weihuahu@stanford.edu
Jure Leskovec Stanford University jure@cs.stanford.edu
Stefanie Jegelka MIT stefje@mit.edu

Abstract

Graph Neural Networks (GNNs) are an effective framework for representation learning of graphs. GNNs follow a neighborhood aggregation scheme, where the representation vector of a node is computed by recursively aggregating and transforming representation vectors of its neighboring nodes. Many GNN variants have been proposed and have achieved state-of-the-art results on both node and graph classification tasks. However, despite GNNs revolutionizing graph representation learning, there is limited understanding of their representational properties and limitations. Here, we present a theoretical framework for analyzing the expressive power of GNNs to capture different graph structures. Our results characterize the discriminative power of popular GNN variants, such as Graph Convolutional Networks and GraphSAGE, and show that they cannot learn to distinguish certain simple graph structures. We then develop a simple architecture that is provably the most expressive among the class of GNNs and is as powerful as the Weisfeiler-Lehman graph isomorphism test. We empirically validate our theoretical findings on a number of graph classification benchmarks, and demonstrate that our model achieves state-of-the-art performance.

Graph G	Graph H	An isomorphism between G and H
		$f(a) = 1$ $f(b) = 6$ $f(c) = 8$ $f(d) = 3$ $f(g) = 5$ $f(h) = 2$ $f(i) = 4$ $f(j) = 7$

同构



不同构

https://arxiv.org/abs/1810.00826

WL(Weisfeler-Lehmann)-test

WL-test 算法步骤：

1. 给图中每个节点按照度值标记，或者给相同的标记
2. 利用HASH函数，将节点已经起邻居标签的multi-set影射为新的标签，直到收敛：

$$l_{\mathcal{G}_i}^{(i)}(v) = \text{HASH}\left(l_{\mathcal{G}_i}^{(i-1)}(v), \left\{\left\{l_{\mathcal{G}_i}^{(i-1)}(u)\ \forall u \in \mathcal{N}(v)\right\}\right\}\right),$$

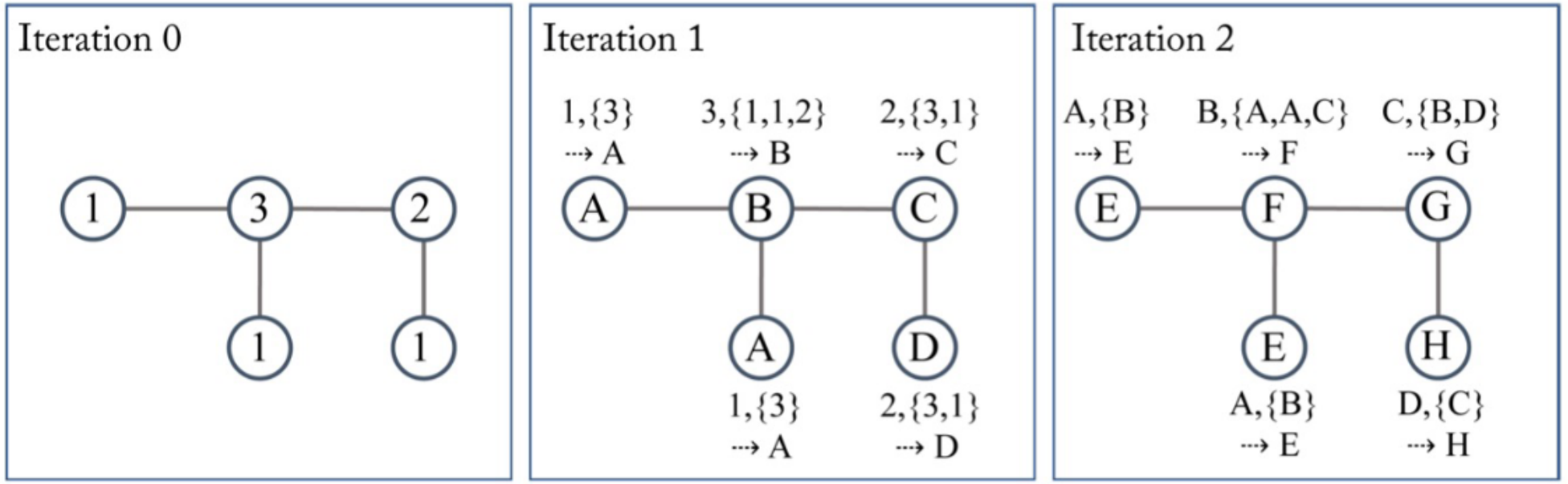
3. 收敛后，统计两个图节点标签的multi-set，如果相同，则同构。

$$L_{\mathcal{G}_j} = \left\{\left\{l_{\mathcal{G}_j}^{(i)}(v), \forall v \in \mathcal{V}_j, i = 0, ..., K-1\right\}\right\}$$

https://arxiv.org/abs/1810.00826

WL-test

WL-test 算法步骤



Graph Isomorphism Network

Published as a conference paper at ICLR 2019

How Powerful are Graph Neural Networks?

Keyulu Xu *†
MIT
keyulu@mit.edu

Weihua Hu *‡
Stanford University
weihuahu@stanford.edu

Jure Leskovec
Stanford University
jure@cs.stanford.edu

Stefanie Jegelka
MIT
stefje@mit.edu

Abstract

Graph Neural Networks (GNNs) are an effective framework for representation learning of graphs. GNNs follow a neighborhood aggregation scheme, where the representation vector of a node is computed by recursively aggregating and transforming representation vectors of its neighboring nodes. Many GNN variants have been proposed and have achieved state-of-the-art results on both node and graph classification tasks. However, despite GNNs revolutionizing graph representation learning, there is limited understanding of their representational properties and limitations. Here, we present a theoretical framework for analyzing the expressive power of GNNs to capture different graph structures. Our results characterize the discriminative power of popular GNN variants, such as Graph Convolutional Networks and GraphSAGE, and show that they cannot learn to distinguish certain simple graph structures. We then develop a simple architecture that is provably the most expressive among the class of GNNs and is as powerful as the Weisfeiler-Lehman graph isomorphism test. We empirically validate our theoretical findings on a number of graph classification benchmarks, and demonstrate that our model achieves state-of-the-art performance.

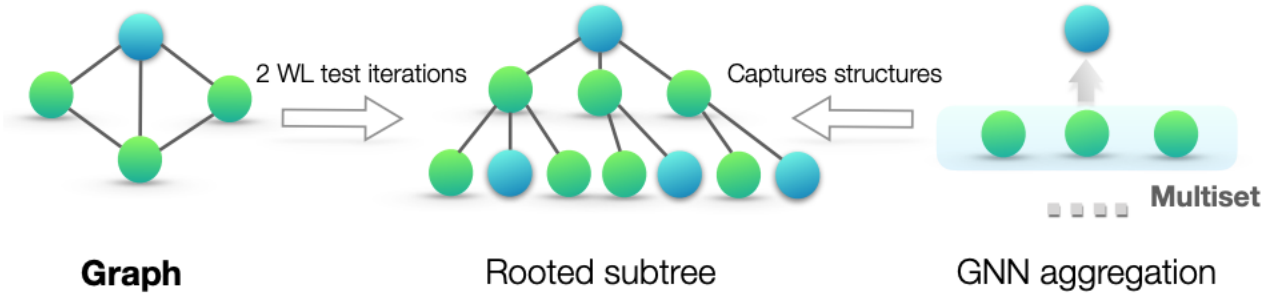
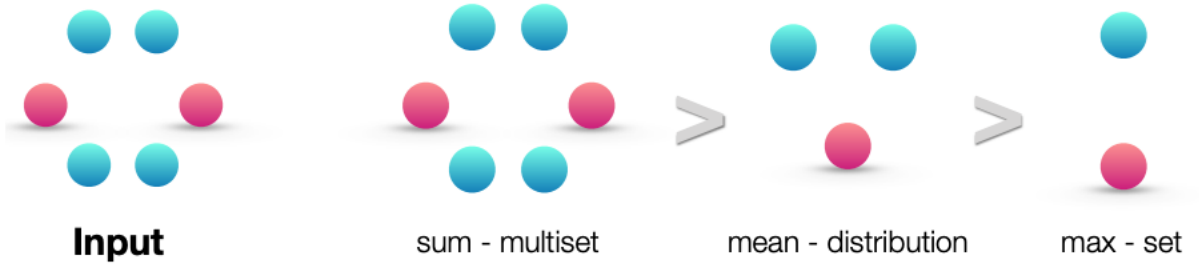


Figure 1: **An overview of our theoretical framework.** Middle panel: rooted subtree structures (at the blue node) that the WL test uses to distinguish different graphs. Right panel: if a GNN's aggregation function captures the *full multiset* of node neighbors, the GNN can capture the rooted subtrees in a recursive manner and be as powerful as the WL test.

$$h_v^{(k)} = \mathrm{MLP}^{(k)}\left(\left(1+\epsilon^{(k)}\right)\cdot h_v^{(k-1)} + \sum_{u\in\mathcal{N}(v)} h_u^{(k-1)}\right)$$



https://arxiv.org/abs/1810.00826

G-GCN

GENERATIVE GRAPH CONVOLUTIONAL NETWORK FOR GROWING GRAPHS

Da Xu, Chuanwei Ruan, Kamiya Motwani, Evren Korpeoglu, Sushant Kumar, Kannan Achan

Walmart Labs, Sunnyvale, California, USA

ABSTRACT

Modeling generative process of growing graphs has wide applications in social networks and recommendation systems, where cold start problem leads to new nodes isolated from existing graph. Despite the emerging literature in learning graph representation and graph generation, most of them can not handle isolated new nodes without nontrivial modifications. The challenge arises due to the fact that learning to generate representations for nodes in observed graph relies heavily on topological features, whereas for new nodes only node attributes are available. Here we propose a unified generative graph convolutional network that learns node representations for all nodes adaptively in a generative model framework, by sampling graph generation sequences constructed from observed graph data. We optimize over a variational lower bound that consists of a graph reconstruction term and an adaptive Kullback-Leibler divergence regularization term. We demonstrate the superior performance of our approach on several benchmark citation network datasets.

Index Terms— Graph representation learning, sequential generative model, variational autoencoder, growing graph

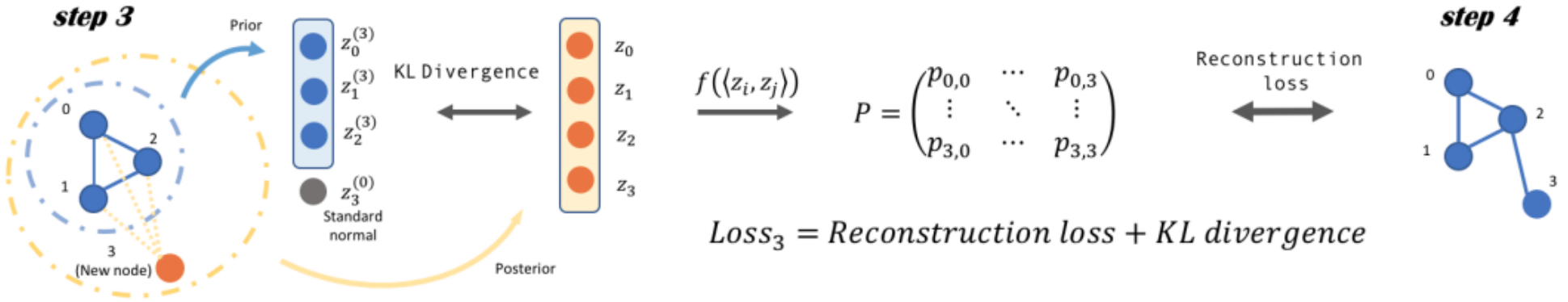
neighborhood information [10]. Variants of *GCN* have been proposed to tackle the computational complexity for large graphs, such as *FastGCN* [11] which applies graph sampling techniques. *GraphSAGE* [12] is another time-efficient inductive graph representation learning approach that implements localized neighborhood aggregations.

On the other side, advancements in generative models compatible with deep neural networks such as variational autoencoders (*VAE*) [13, 14] and generative adversarial networks (*GAN*) [15] have enabled direct modeling for generation of complex distributions. As a consequence, there have been several recent work on deep generative models for graphs [16, 17, 18, 19]. However, many of them only deal with fixed graphs [19, 18] or graphs of very small sizes [17, 1]. Moreover, most graph representation learning methods require at least some topological features from all nodes in order to conduct neighborhood aggregations or random walks, which is clearly infeasible for growing graphs with isolated new nodes. To obtain embeddings and further generate graph structures for both new and old nodes, it is essential to utilize node attributes. Also, instead of learning how the observed graph is generated as a whole, the graph generation

https://arxiv.org/pdf/1903.02640.pdf

G-GCN

Fig. 1. An illustration for the workflow of our approach at a transition step. The growing graph has an observed subgraph with three nodes (node 0, 1 and 2) and an incoming new node (node 3). The informative conditional priors for latent factors $z_{0:2}^{(3)}$ contain structural information of the observed subgraph. The encoding distributions for all latent factors are formed according to the candidate adjacency matrix where candidate edges (the dashed edges) are added to original subgraph.



Encoding:

$$\boldsymbol{\mu}(\boldsymbol{z}^i|\boldsymbol{X}_{\leq i+1}) = \hat{\tilde{\boldsymbol{A}}}_{\leq i+1}\sigma(\hat{\tilde{\boldsymbol{A}}}_{\leq i+1}\boldsymbol{X}_{\leq i+1}\boldsymbol{W}_0)\boldsymbol{W}_1,$$
$$\mathrm{diag}(\boldsymbol{\Sigma}(\boldsymbol{z}^i|\boldsymbol{X}_{\leq i+1}) = \hat{\tilde{\boldsymbol{A}}}_{\leq i+1}\sigma(\hat{\tilde{\boldsymbol{A}}}_{\leq i+1}\boldsymbol{X}_{\leq i+1}\boldsymbol{W}_0)\boldsymbol{W}_2,$$

Decoding:

$$p_{i,j} = p(\boldsymbol{A}_{i,j} = 1|\boldsymbol{z}_i, \boldsymbol{z}_j) = f(\langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle)$$

$$(\tilde{\boldsymbol{A}}_{t+1}^{\pi})_{t+1,t+1} = 1, (\tilde{\boldsymbol{A}}_{t+1}^{\pi})_{1:t,1:t} = \boldsymbol{A}_t^{\pi},$$
$$p((\tilde{\boldsymbol{A}}_{t+1}^{\pi})_{k,t+1} = 1) = \tilde{p} \text{ for } k = 1, 2, \cdots, t.$$

Prior:

$$p_0^i(\boldsymbol{z}_{1:i}^{i+1}|\boldsymbol{A}_{\leq i}, \boldsymbol{X}_{\leq i}) = p_\phi(\boldsymbol{z}_{1:i}|\tilde{\boldsymbol{A}}_{\leq i+1}, \boldsymbol{X}_{\leq i+1})$$
$$\boldsymbol{z}_{i+1}^{i+1}|\boldsymbol{A}_{\leq i}, \boldsymbol{X}_{\leq i} \sim N(0, I)$$

Loss:

$$\log p(\boldsymbol{A}_{\leq i+1}, \boldsymbol{X}_{\leq i+1}|\boldsymbol{A}_{\leq i}, \boldsymbol{X}_{\leq i})$$
$$\geq E_{q_\phi^i(\boldsymbol{z}^{i+1}|\tilde{\boldsymbol{A}}_{\leq i+1}, \boldsymbol{X}_{\leq i+1})}[\log p_\theta^i(\boldsymbol{A}_{\leq i}|\boldsymbol{z}^i)]$$
$$- KL(q_\phi^i(\boldsymbol{z}^{i+1}|\tilde{\boldsymbol{A}}_{\leq i+1}, \boldsymbol{X}_{\leq i+1})\|p_0^i(\boldsymbol{z}^{i+1}|\boldsymbol{A}_{\leq i}, \boldsymbol{X}_{\leq i})) + C$$
$$\equiv ELBO_i + C.$$

https://arxiv.org/pdf/1903.02640.pdf

Deep NC Method

DeepNC: Deep Generative Network Completion

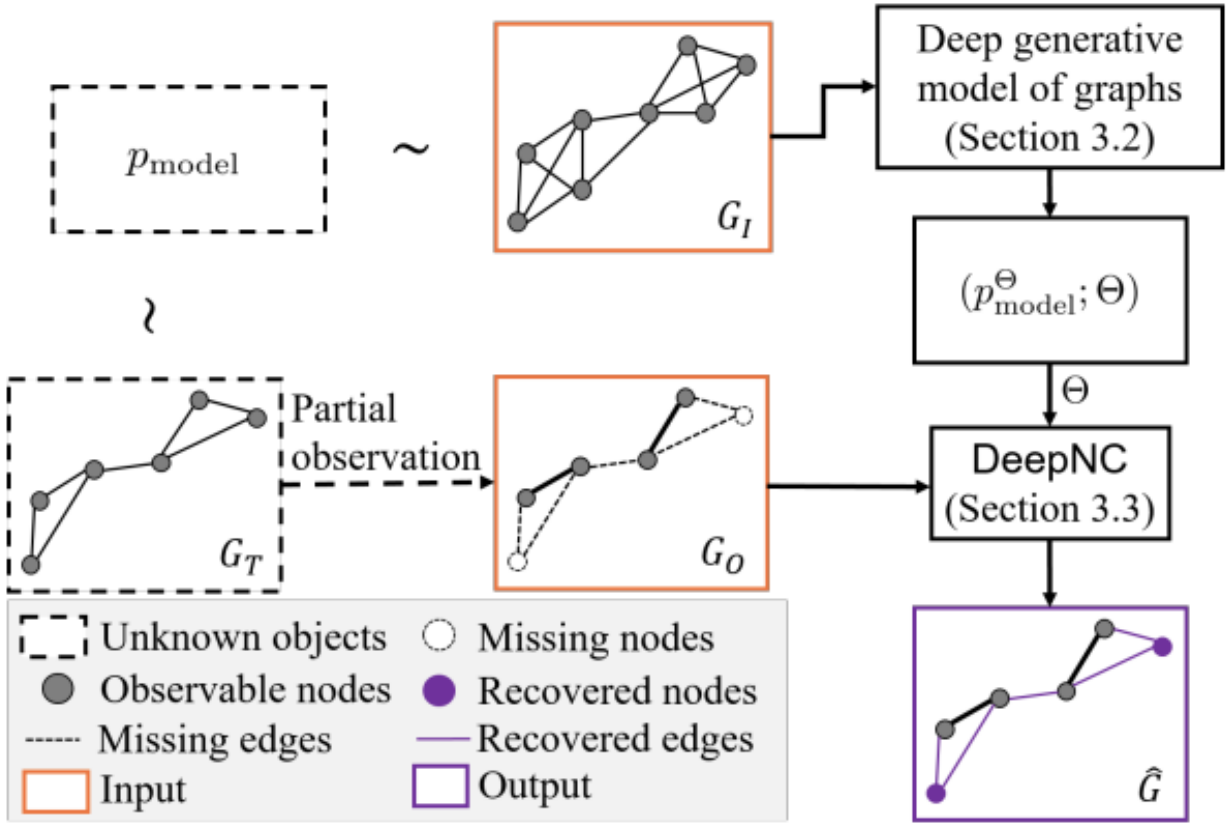
Cong Tran, *Student Member, IEEE*, Won-Yong Shin, *Senior Member, IEEE*, Andreas Spitz, and Michael Gertz

Abstract—Most network data are collected from partially observable networks with both missing nodes and missing edges, for example, due to limited resources and privacy settings specified by users on social media. Thus, it stands to reason that inferring the missing parts of the networks by performing *network completion* should precede downstream applications. However, despite this need, the recovery of missing nodes and edges in such incomplete networks is an insufficiently explored problem due to the modeling difficulty, which is much more challenging than link prediction that only infers missing edges. In this paper, we present DeepNC, a novel method for inferring the missing parts of a network based on a *deep generative* model of graphs. Specifically, our method first learns a likelihood over edges via an *autoregressive generative* model, and then identifies the graph that maximizes the learned likelihood conditioned on the observable graph topology. Moreover, we propose a computationally efficient DeepNC algorithm that *consecutively* finds individual nodes that maximize the probability in each node generation step, as well as an enhanced version using the expectation-maximization algorithm. The runtime complexities of both algorithms are shown to be *almost linear* in the number of nodes in the network. We empirically demonstrate the superiority of DeepNC over state-of-the-art network completion approaches.

Index Terms—Autoregressive generative model; deep generative model of graphs; inference; network completion; partially observable network

C. Tran, W.-Y. Shin, A. Spitz and M. Gertz, "DeepNC: Deep generative network completion", *IEEE Trans. Pattern Analysis Machine Intell*.

Deep NC Method



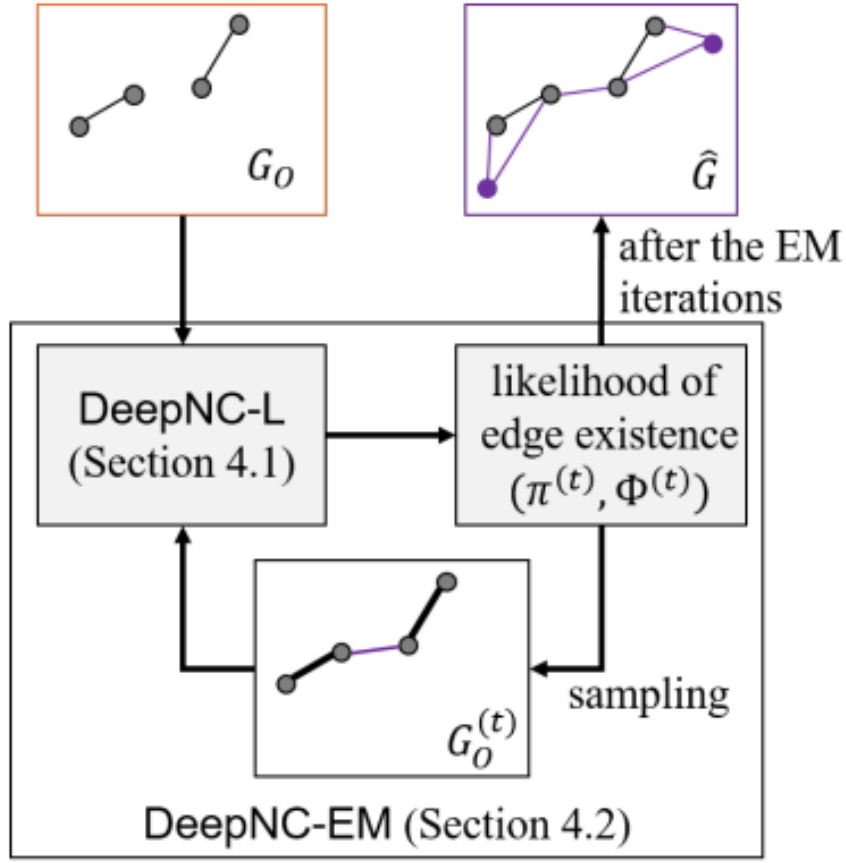


Fig. 1: The schematic overview of our DeepNC method.

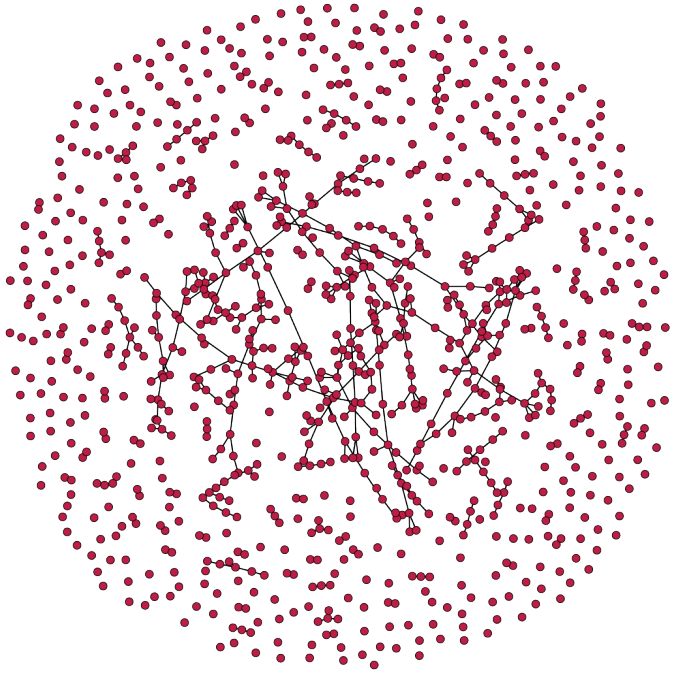
C. Tran, W.-Y. Shin, A. Spitz and M. Gertz, "DeepNC: Deep generative network completion", *IEEE Trans. Pattern Analysis Machine Intell*.

Outline

- Background
- Network Embedding
- Graph Neural Networks
  - Graph Convolution
  - Graph Attention Network
  - Other extension
- Variational Graph Autoencoder
- Graph Generation

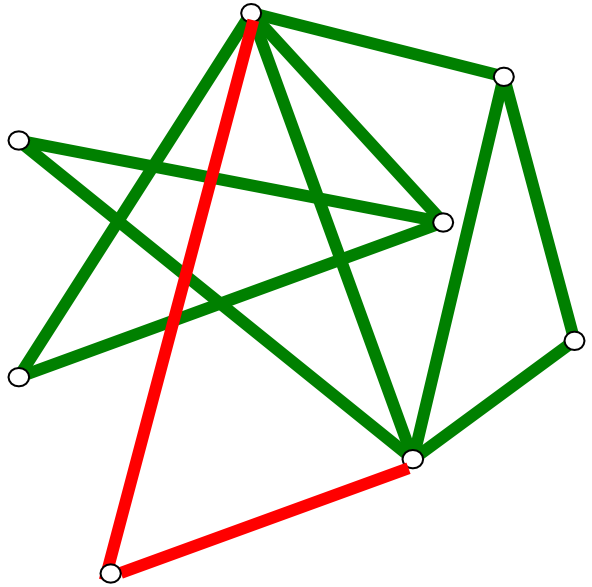
Generative Models of Graphs

ER model



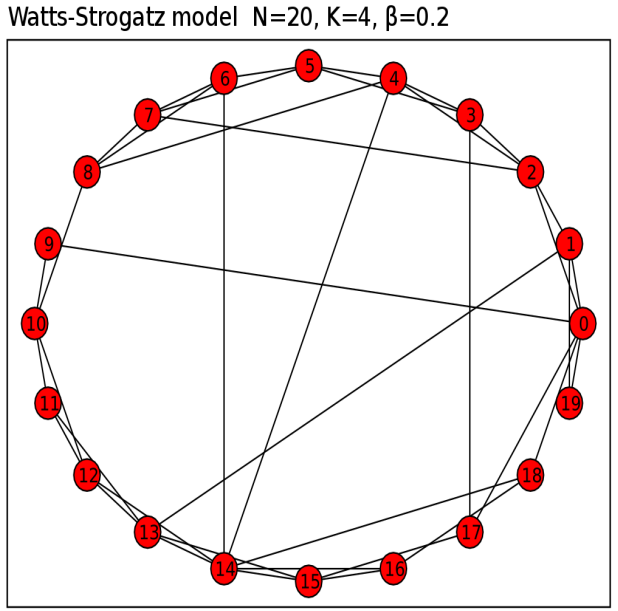
$$p^M (1-p)^{\binom{n}{2}-M}$$

BA Model



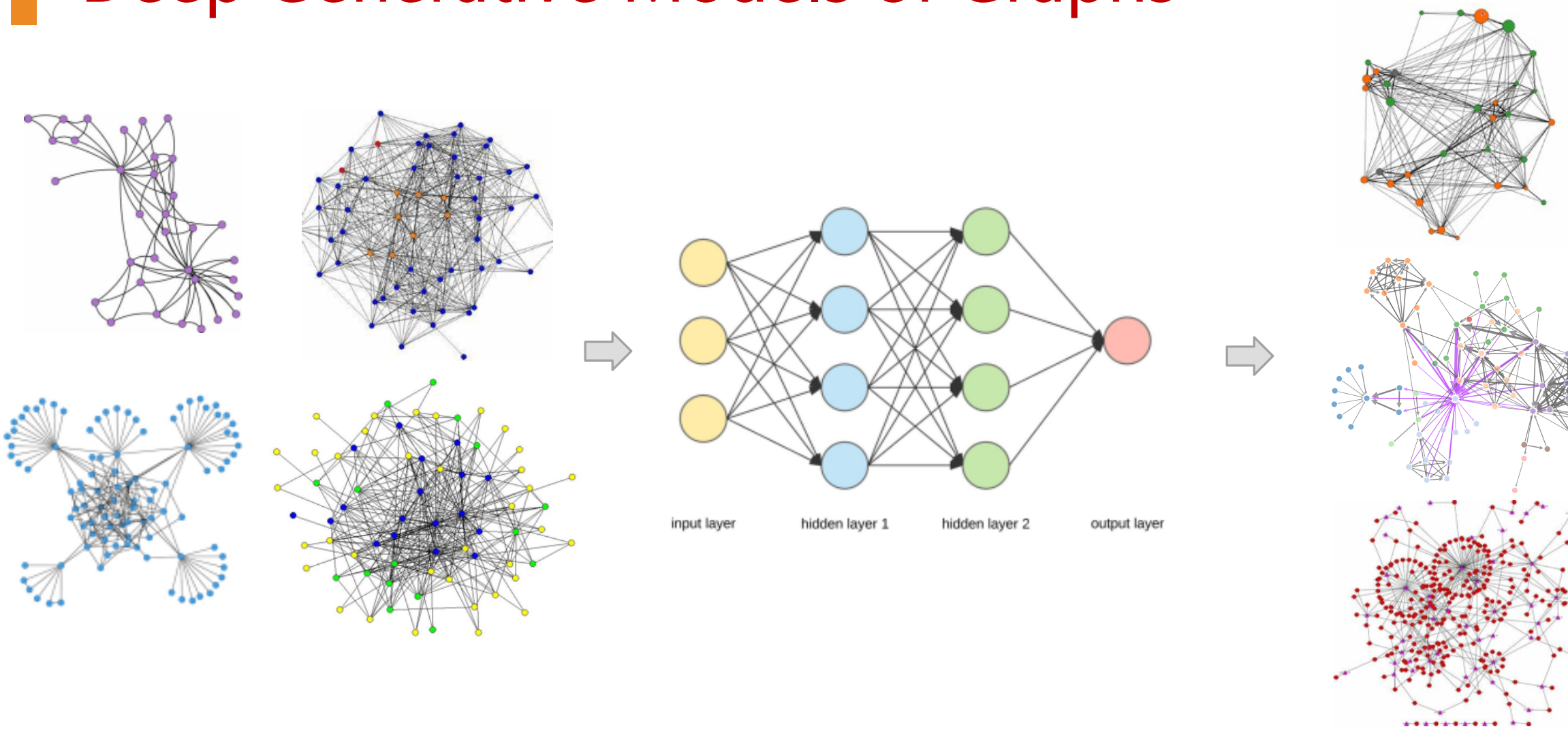
$$\Pi(k) = \frac{k_i}{\sum_i k_i}$$

WS Model



1. Construct a regular ring lattice, a graph with N nodes each connected to K neighbors, $K/2$ on each side. That is, if the nodes are labeled $0 \ldots N-1$, there is an edge (i,j) if and only if $0 < |i-j| \bmod \left(N - 1 - \frac{K}{2}\right) \leq \frac{K}{2}$.
2. For every node $i = 0, \ldots, N-1$ take every edge connecting i to its $K/2$ rightmost neighbors, that is every edge $(i, j \bmod N)$ with $i < j \leq i + K/2$, and rewire it with probability β. Rewiring is done by replacing $(i, j \bmod N)$ with (i,k) where k is chosen uniformly at random from all possible nodes while avoiding self-loops ($k \neq i$) and link duplication (there is no edge (i,k') with $k' = k$ at this point in the algorithm).

Deep Generative Models of Graphs



Deep Generative Models of Graphs

A Systematic Survey on Deep Generative Models for Graph Generation

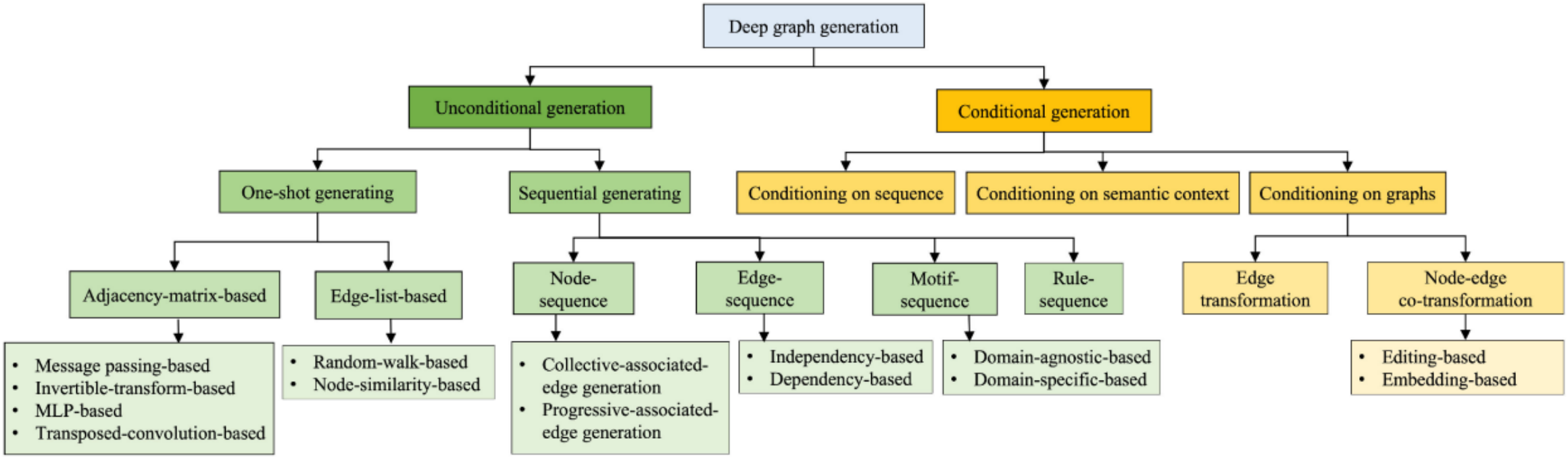
Xiaojie Guo and Liang Zhao

Abstract—Graphs are important data representations for describing objects and their relationships, which appear in a wide diversity of real-world scenarios. As one of a critical problem in this area, graph generation considers learning the distributions of given graphs and generating more novel graphs. Owing to their wide range of applications, generative models for graphs, which have a rich history, however, are traditionally hand-crafted and only capable of modeling a few statistical properties of graphs. Recent advances in deep generative models for graph generation is an important step towards improving the fidelity of generated graphs and paves the way for new kinds of applications. This article provides an extensive overview of the literature in the field of deep generative models for graph generation. First, the formal definition of deep generative models for the graph generation and the preliminary knowledge are provided. Second, taxonomies of deep generative models for both unconditional and conditional graph generation are proposed respectively; the existing works of each are compared and analyzed. After that, an overview of the evaluation metrics in this specific domain is provided. Finally, the applications that deep graph generation enables are summarized and five promising future research directions are highlighted.

Index Terms—Graph generation, graph neural network, deep generative models for graphs

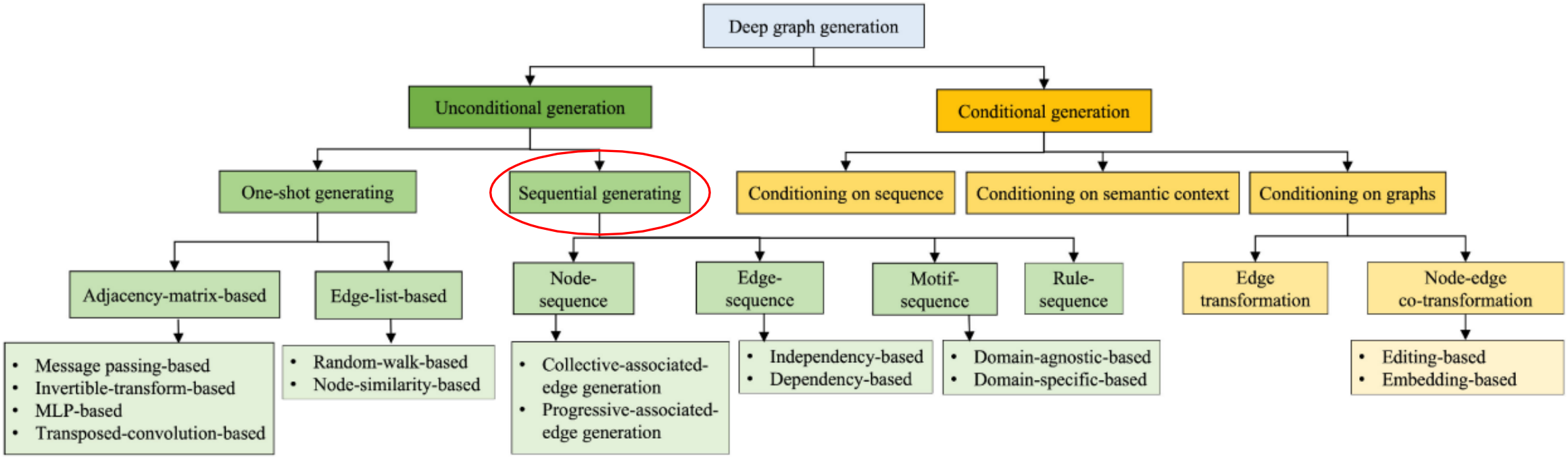
https://ieeexplore.ieee.org/document/9920219

Deep Generative Models of Graphs



https://ieeexplore.ieee.org/document/9920219

Deep Generative Models of Graphs



https://ieeexplore.ieee.org/document/9920219

Graph RNN

Jure Leskovec

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]

Abstract

Modeling and generating graphs is fundamental for studying networks in biology, engineering, and social sciences. However, modeling complex distributions over graphs and then efficiently sampling from these distributions is challenging due to the non-unique, high-dimensional nature of graphs and the complex, non-local dependencies that exist between edges in a given graph. Here we propose GraphRNN, a deep autoregressive model that addresses the above challenges and approximates any distribution of graphs with minimal assumptions about their structure. GraphRNN learns to generate graphs by training on a representative set of graphs and decomposes the graph generation process into a sequence of node and edge formations, conditioned on the graph structure generated so far. In order to quantitatively evaluate the performance of GraphRNN, we introduce a benchmark suite of datasets, baselines and novel evaluation metrics based on Maximum Mean Discrepancy, which measure distances between sets of graphs. Our experiments show that GraphRNN significantly outperforms all baselines, learning to generate diverse graphs that match the structural characteristics of a target set, while also scaling to graphs $50\times$ larger than previous deep models.

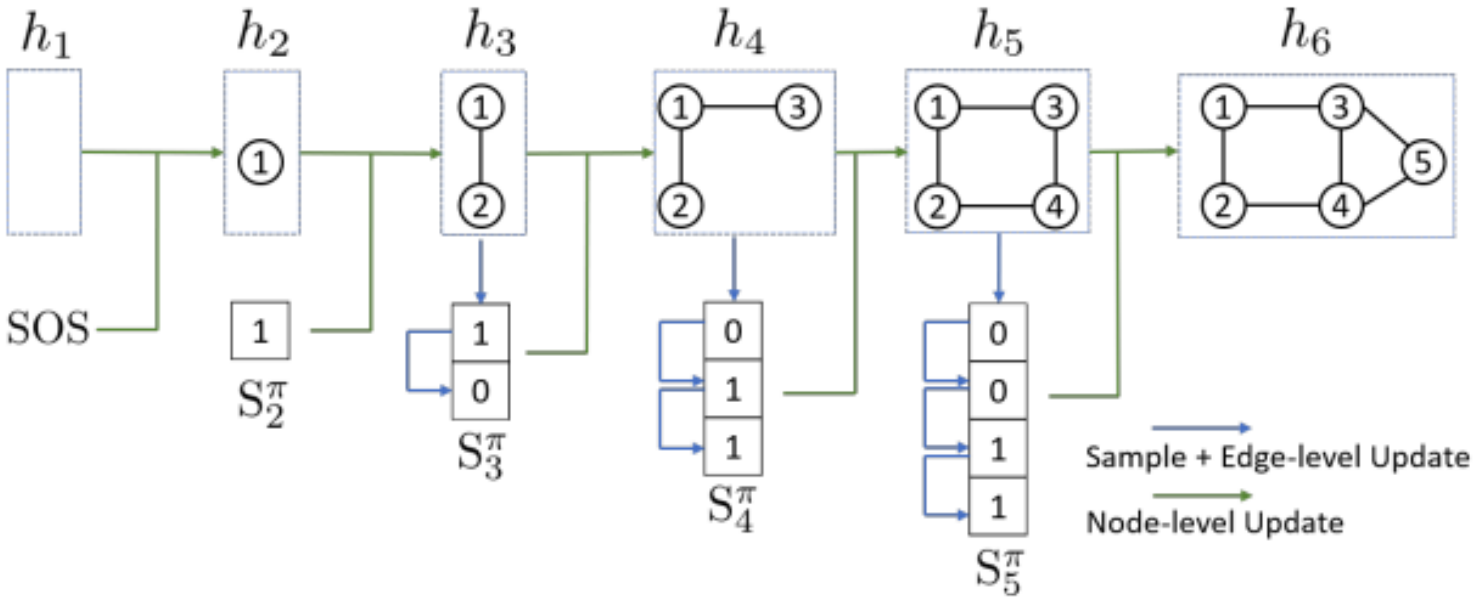
graphs based on a priori structural assumptions (Newman, 2010). However, a key open challenge in this area is developing methods that can directly *learn* generative models from an observed set of graphs. Developing generative models that can learn directly from data is an important step towards improving the fidelity of generated graphs, and paves a way for new kinds of applications, such as discovering new graph structures and completing evolving graphs.

In contrast, traditional generative models for graphs (*e.g.*, Barabási-Albert model, Kronecker graphs, exponential random graphs, and stochastic block models) (Erdős & Rényi, 1959; Leskovec et al., 2010; Albert & Barabási, 2002; Airoldi et al., 2008; Leskovec et al., 2007; Robins et al., 2007) are hand-engineered to model a particular family of graphs, and thus do not have the capacity to directly learn the generative model from observed data. For example, the Barabási-Albert model is carefully designed to capture the scale-free nature of empirical degree distributions, but fails to capture many other aspects of real-world graphs, such as community structure.

Recent advances in deep generative models, such as variational autoencoders (VAE) (Kingma & Welling, 2014) and generative adversarial networks (GAN) (Goodfellow et al., 2014), have made important progress towards generative modeling for complex domains, such as image and text data. Building on these approaches a number of deep learning models for generating graphs have been proposed (Kipf & Welling, 2016; Grover et al., 2017; Simonovsky & Ko-

https://arxiv.org/pdf/1802.08773.pdf

Graph RNN



$$S^\pi = f_S(G, \pi) = (S_1^\pi, ..., S_n^\pi),$$

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

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

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

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

$$S^\pi = f_S(G, \mathbf{BFS}(G, \pi)),$$

https://arxiv.org/pdf/1802.08773.pdf

Graph RNN

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}

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

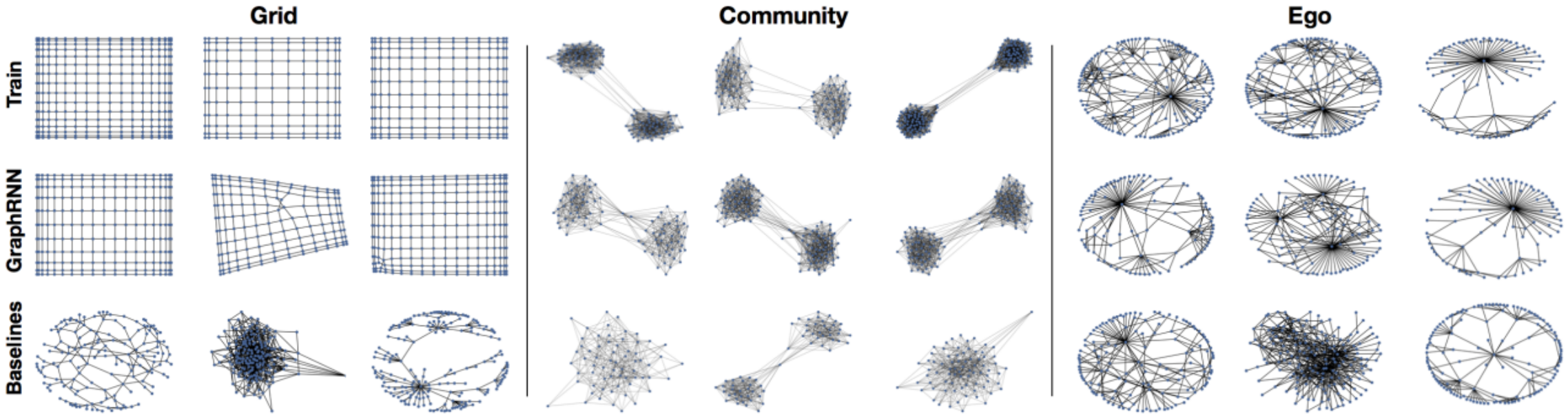
$S_i^\pi \sim \mathcal{P}_{\theta_i}$ {sample node i's edge connections}

until S_i^π is EOS

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

https://arxiv.org/pdf/1802.08773.pdf

Graph RNN - Results



https://arxiv.org/pdf/1802.08773.pdf

Graph RNN - Results

Table 2. GraphRNN compared to state-of-the-art deep graph generative models on small graph datasets using MMD and negative log-likelihood (NLL). $(\max(|V|), \max(|E|))$ of each dataset is shown. (DeepVAE and GraphVAE cannot scale to the graphs in Table 1.)

	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

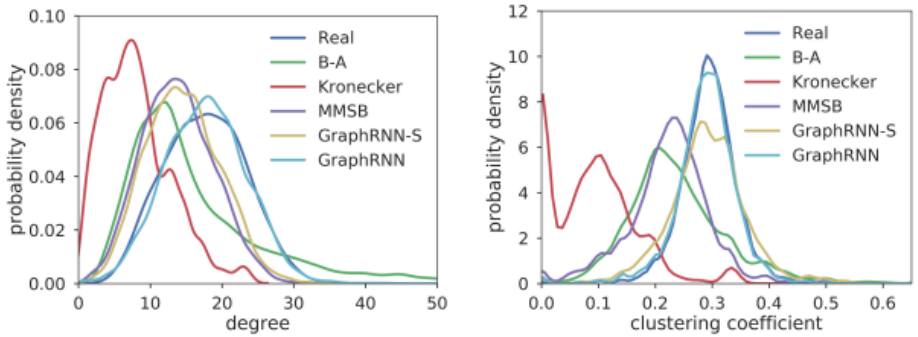


Figure 3. Average degree (Left) and clustering coefficient (Right) distributions of graphs from test set and graphs generated by GraphRNN and baseline models.

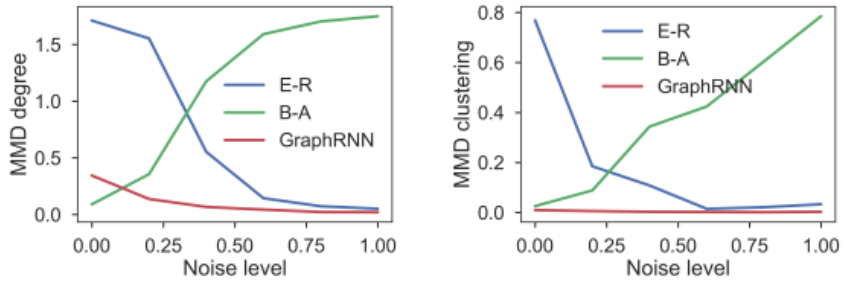


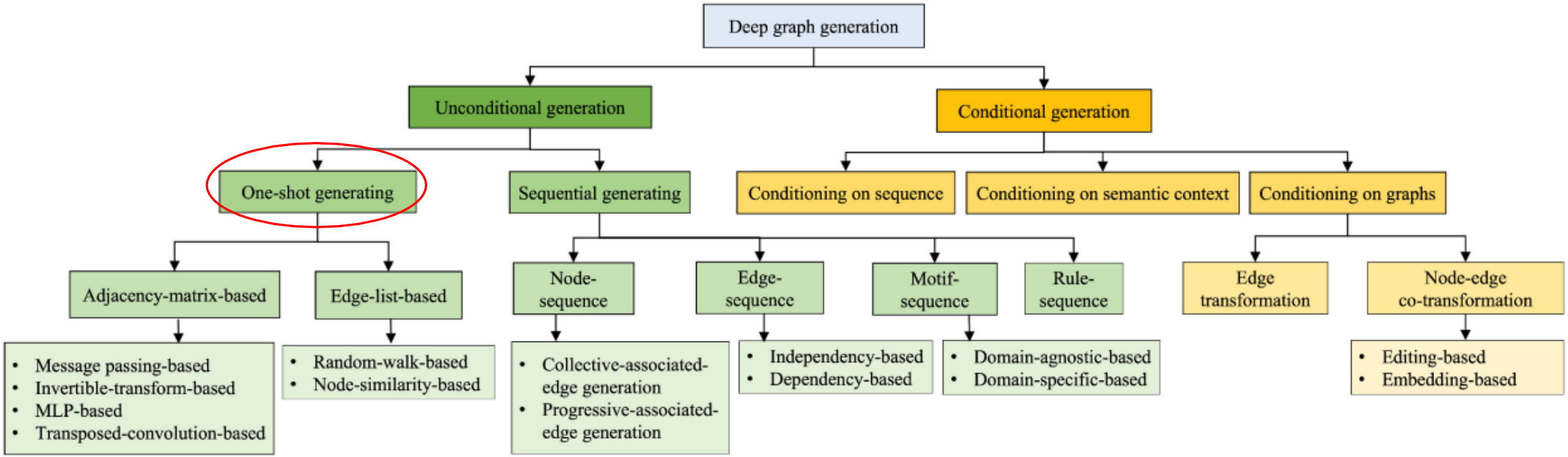
Figure 4. MMD performance of different approaches on degree (Left) and clustering coefficient (Right) under different noise level.

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

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

https://arxiv.org/pdf/1802.08773.pdf

Deep Generative Models of Graphs



https://ieeexplore.ieee.org/document/9920219

Deep Generative Models of Graphs

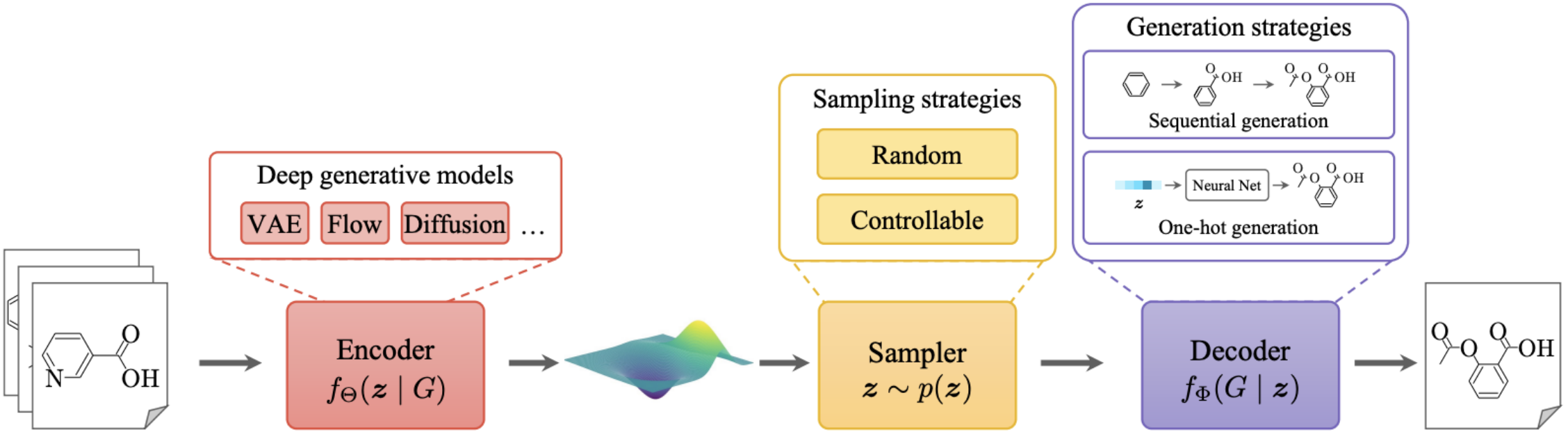


Figure 1: An overview of deep graph generation approaches: the encoder maps observed graphs into a stochastic distribution; the sampler draws latent representations from that distribution; the decoder receives latent codes and produces graphs.

https://arxiv.org/pdf/2203.06714.pdf

Graph Normalizing Flows

Graph Normalizing Flows

Jenny Liu*
University of Toronto
Vector Institute
jyliu@cs.toronto.edu

Aviral Kumar*†
UC Berkeley
aviralk@berkeley.edu

Jimmy Ba
University of Toronto
Vector Institute
jba@cs.toronto.edu

Jamie Kiros
Google Research
kiros@google.com

Kevin Swersky
Google Research
kswersky@google.com

Abstract

We introduce graph normalizing flows: a new, reversible graph neural network model for prediction and generation. On supervised tasks, graph normalizing flows perform similarly to message passing neural networks, but at a significantly reduced memory footprint, allowing them to scale to larger graphs. In the unsupervised case, we combine graph normalizing flows with a novel graph auto-encoder to create a generative model of graph structures. Our model is permutation-invariant, generating entire graphs with a single feed-forward pass, and achieves competitive results with the state-of-the art auto-regressive models, while being better suited to parallel computing architectures.

https://arxiv.org/pdf/1905.13177.pdf

Flows: main idea

Invertible function Invertible function Invertible function

x → [ ] → [ ] → ... → [ ] → $p_\theta(x)$

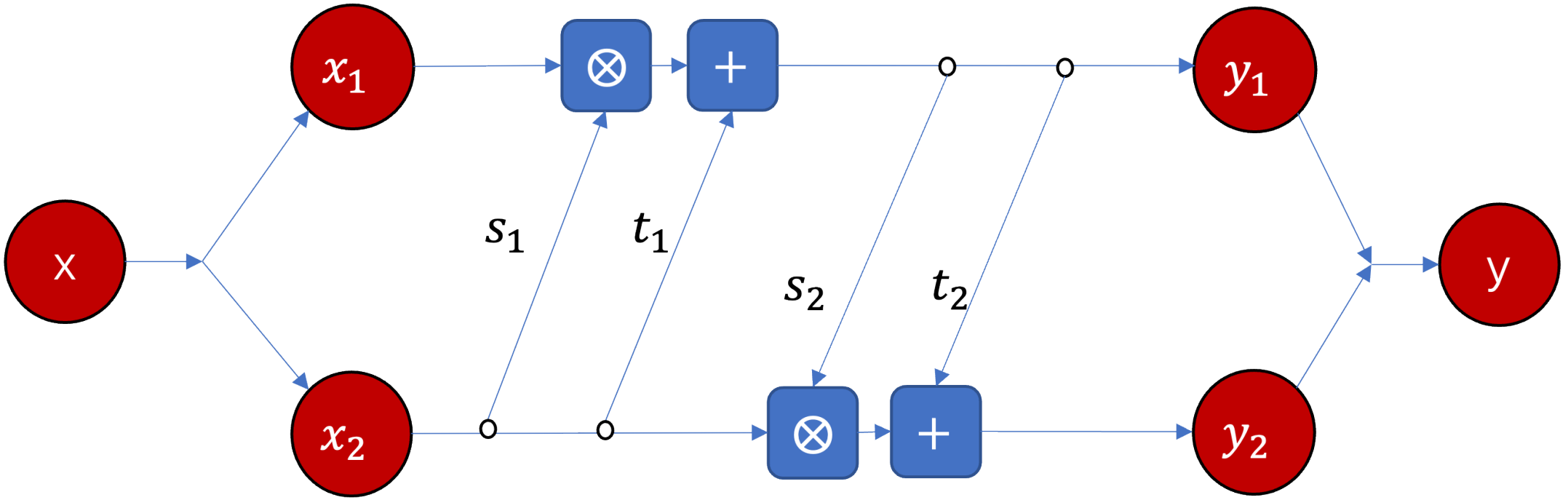
$$z = f_\theta(x)$$

Generally: $z \sim p_Z(z)$

Normalizing Flow: $z \sim \mathcal{N}(0, 1)$

How to train? How to evaluate $p_\theta(x)$? How to sample?

Invertible Neural Network - RealNVP

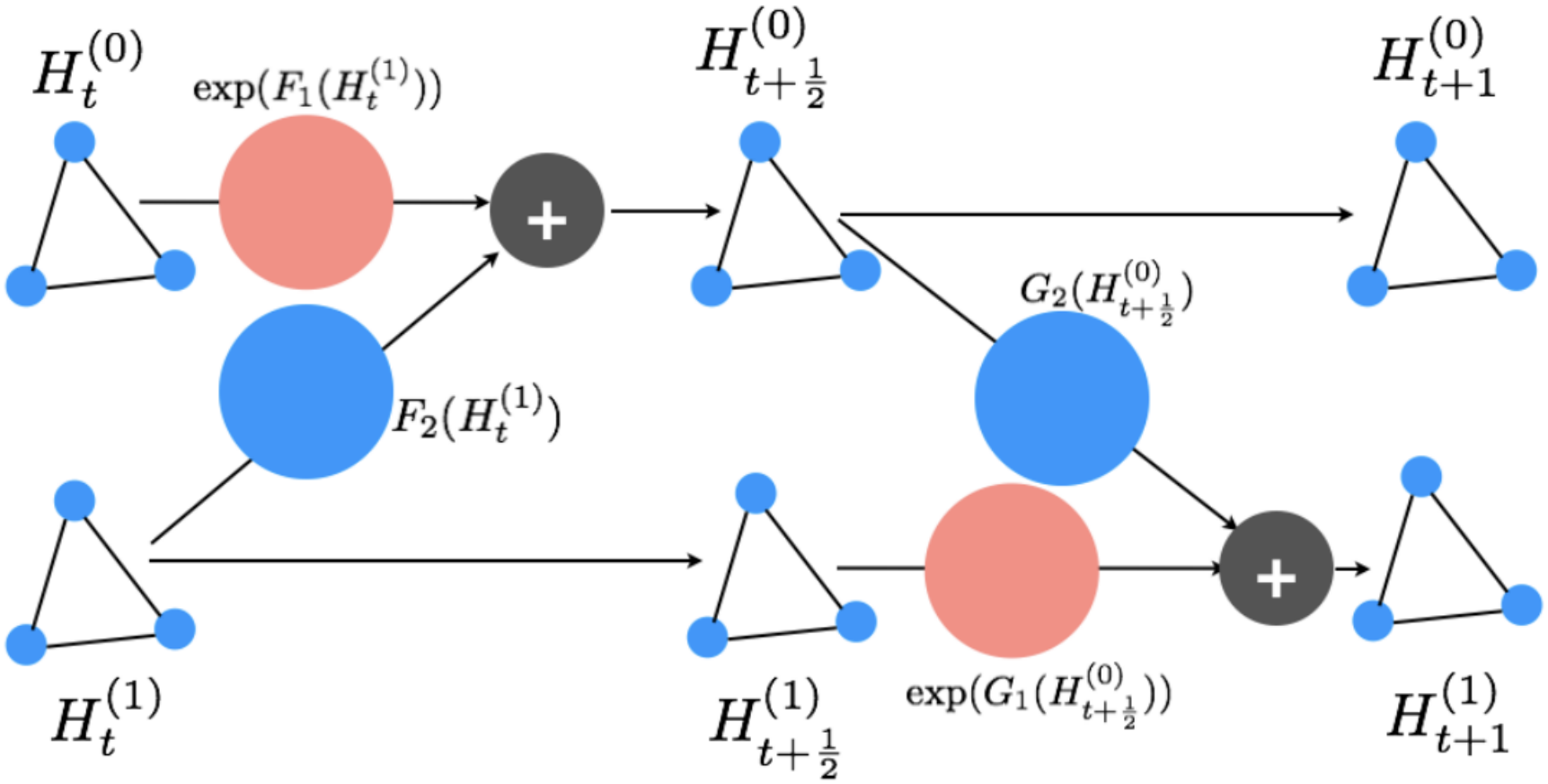


$$y_1 = x_1 \otimes \exp(s_1(x_2)) + t_1(x_2)$$
$$y_2 = x_2 \otimes \exp(s_2(y_1)) + t_2(y_1)$$

$$x_2 = (y_2 - t_2(y_1)) \otimes \exp(-s_2(y_1))$$
$$x_1 = (y_1 - t_1(x_2)) \otimes \exp(-s_1(x_2))$$

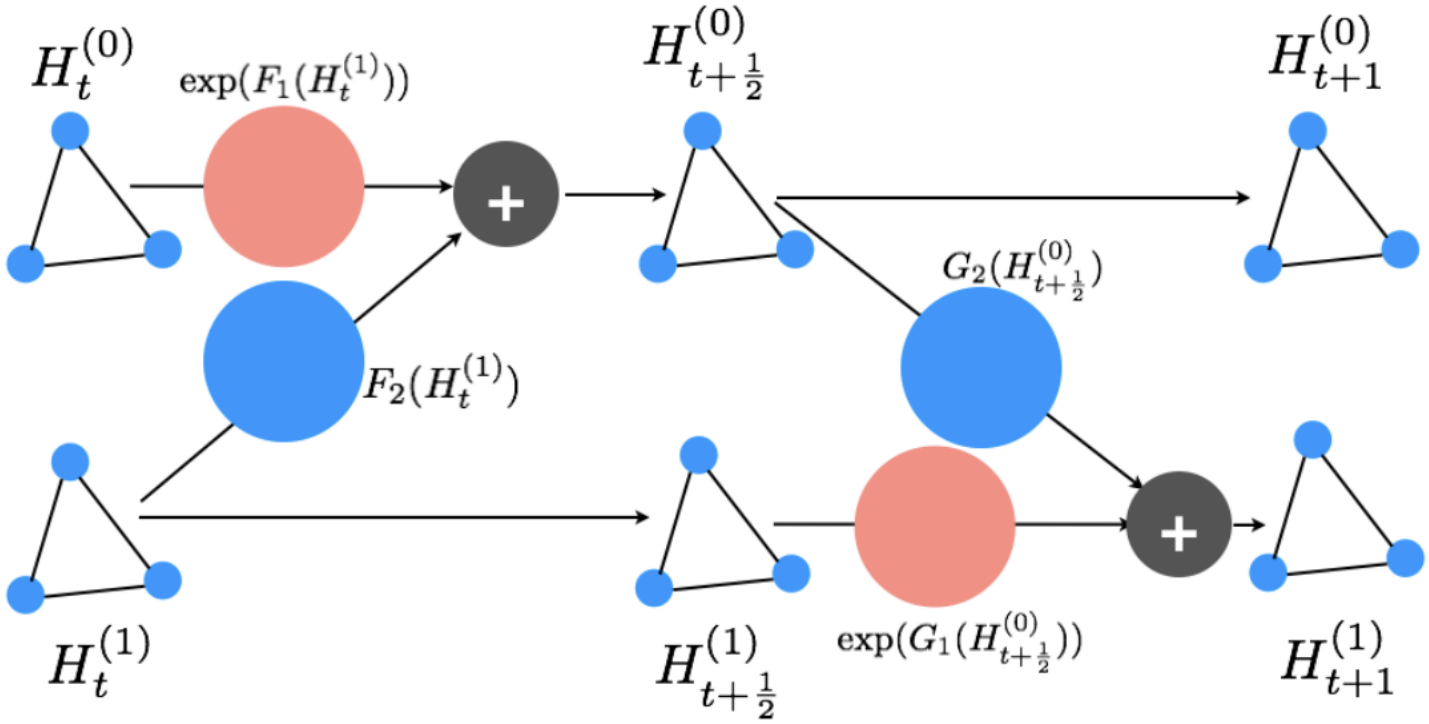
https://arxiv.org/pdf/1605.08803.pdf， DENSITY ESTIMATION USING REAL NVP

Reversible Graph Neural Networks(GRevNets)



https://arxiv.org/pdf/1905.13177.pdf

Reversible Graph Neural Networks(GRevNets)

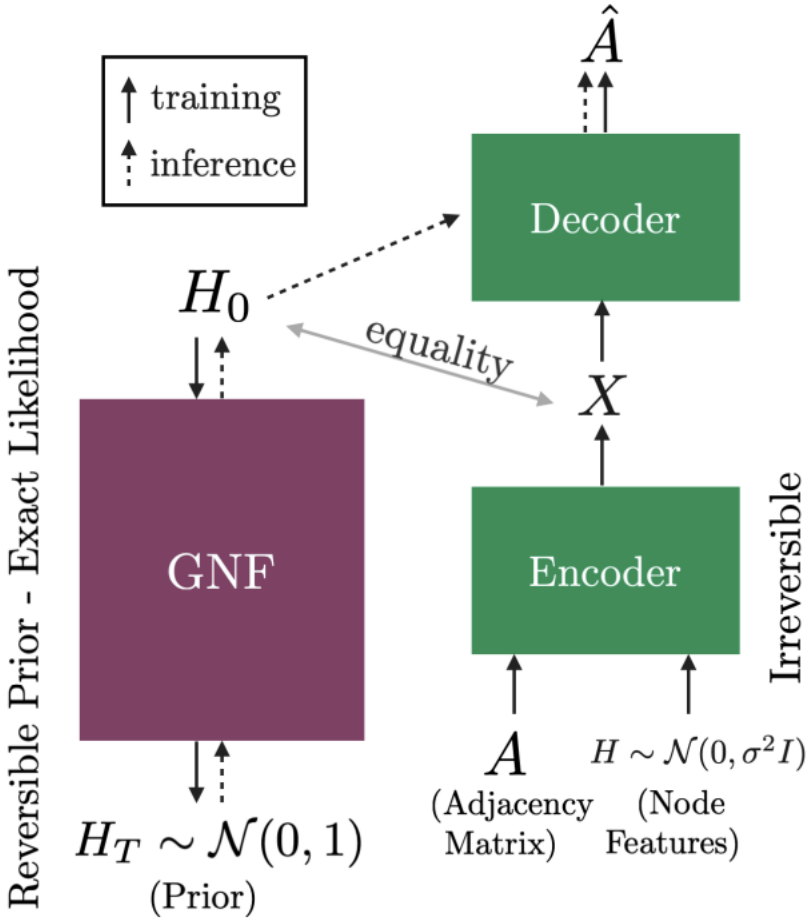


$$H_{t+\frac{1}{2}}^{(0)} = H_t^{(0)} \odot \exp\left(F_1\left(H_t^{(1)}\right)\right) + F_2(H_t^{(1)}) \qquad H_{t+1}^{(0)} = H_{t+\frac{1}{2}}^{(0)}$$

$$H_{t+\frac{1}{2}}^{(1)} = H_t^{(1)} \qquad H_{t+1}^{(1)} = H_{t+\frac{1}{2}}^{(1)} \odot \exp\left(G_1\left(H_{t+\frac{1}{2}}^{(0)}\right)\right) + G_2\left(H_{t+\frac{1}{2}}^{(0)}\right)$$

https://arxiv.org/pdf/1905.13177.pdf

Complete Generation Pipeline



$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i=1}^{N}\sum_{j=1}^{\frac{N}{2}} A_{ij}\log(\hat{A}_{ij}) + (1 - A_{ij})\log(1 - \hat{A}_{ij}).$$

$$\hat{A}_{ij} = \frac{1}{1 + \exp(C(\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 - 1))}$$

https://arxiv.org/pdf/1905.13177.pdf

Results – supervised learning

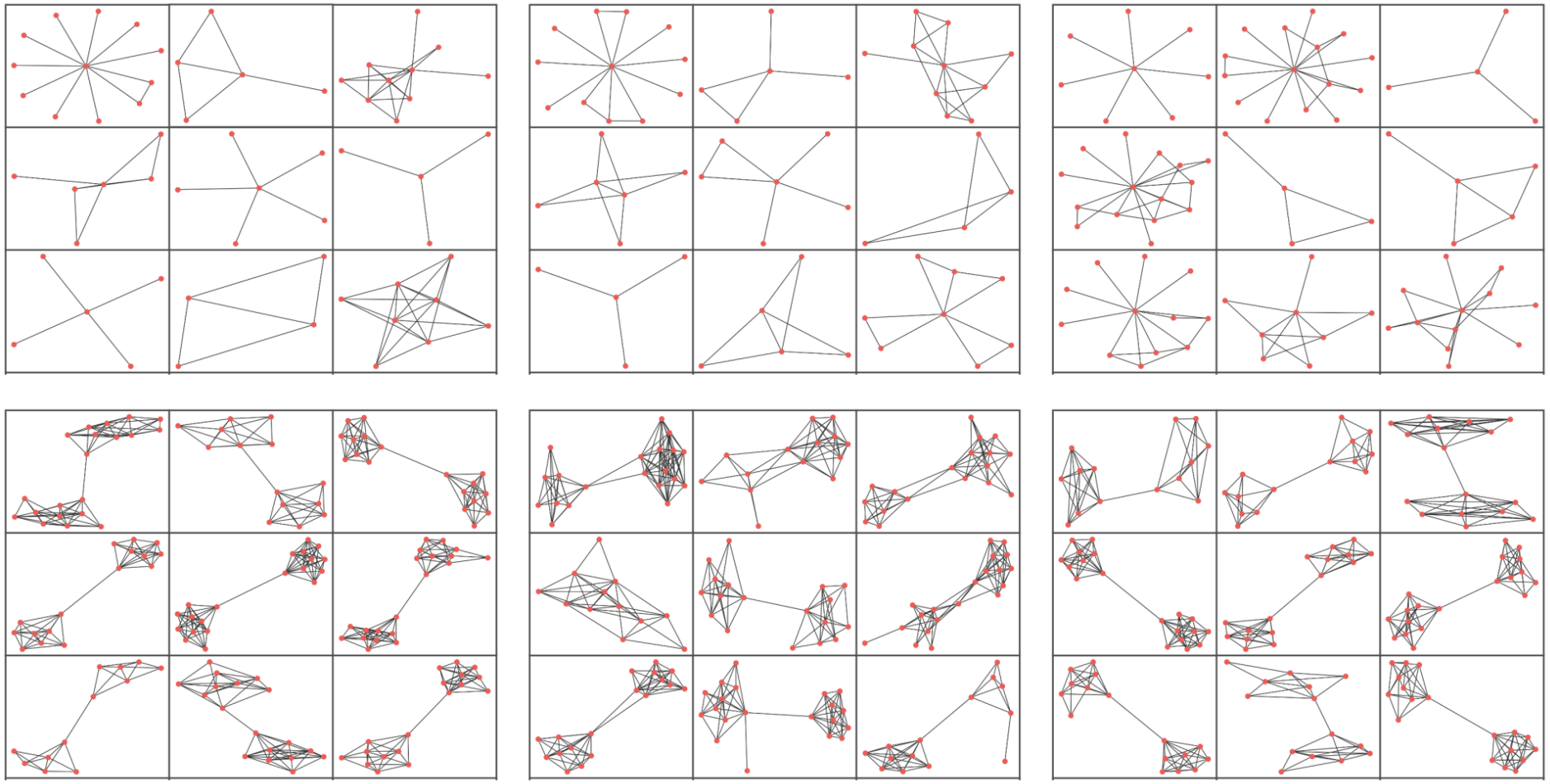
Dataset/Task		GNN	GRevNet	Neumann RBP
Cora	Semi-Supervised	71.9	**74.5**	56.5
Cora	(1% Train)	55.5	**55.8**	54.6
Pubmed	Semi-Supervised	**76.3**	76.0	62.4
Pubmed	(1% Train)	76.6	**77.0**	58.5
PPI	Inductive	**0.78**	0.76	0.70

Model	mu	alpha	HOMO	LUMO	gap	R2
GNN	0.474	0.421	**0.097**	0.124	0.170	27.150
GrevNet	**0.462**	**0.414**	0.098	0.124	**0.169**	**26.380**

Model	ZPVE	U0	U	H	G	Cv
GNN	0.035	0.410	**0.396**	**0.381**	0.373	0.198
GrevNet	0.036	**0.390**	0.407	0.418	**0.359**	**0.195**

https://arxiv.org/pdf/1905.13177.pdf

Results – graph generation



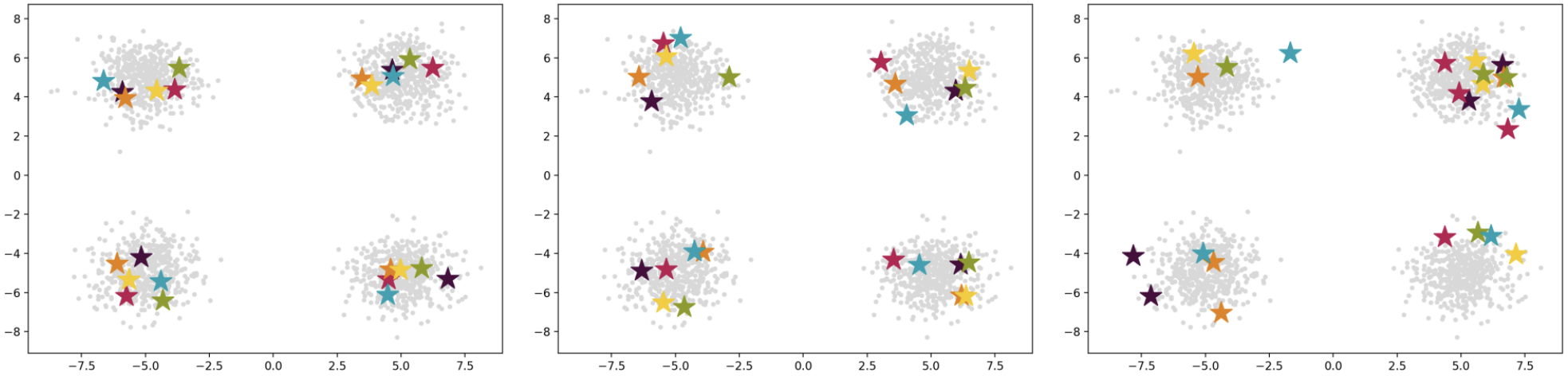
(a) Training data (b) GNF samples (c) GRAPHRNN samples

https://arxiv.org/pdf/1905.13177.pdf

Results – graph generation

MODEL	MOG (NLL)	MOG RING (NLL)	6-HALF MOONS (NLL)
REALNVP	4.2	5.2	-1.2
GNF	**3.6**	**4.2**	**-1.7**

Table 2: Per-node negative log likelihoods (NLL) on synthetic datasets for REALNVP and GNF.



(a) Training examples (b) GNF samples (c) RealNVP samples

https://arxiv.org/pdf/1905.13177.pdf

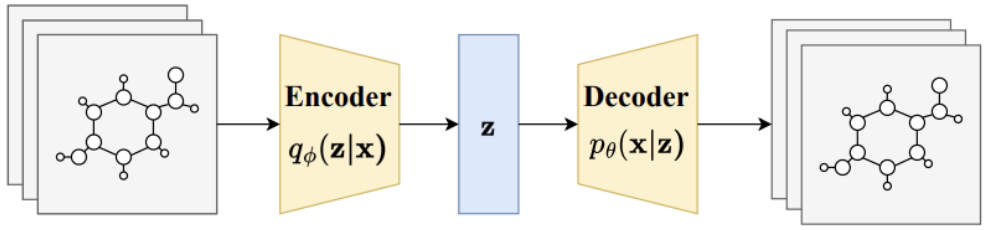
Results – graph generation

	Binary CE		Total # Incorrect Edges		Total # Edges	
Dataset	Train	Test	Train	Test	Train	Test
ego-small	9.8e-4	11e-04	24	32	3758	984
community-small	5e-4	7e-04	10	2	1329	353

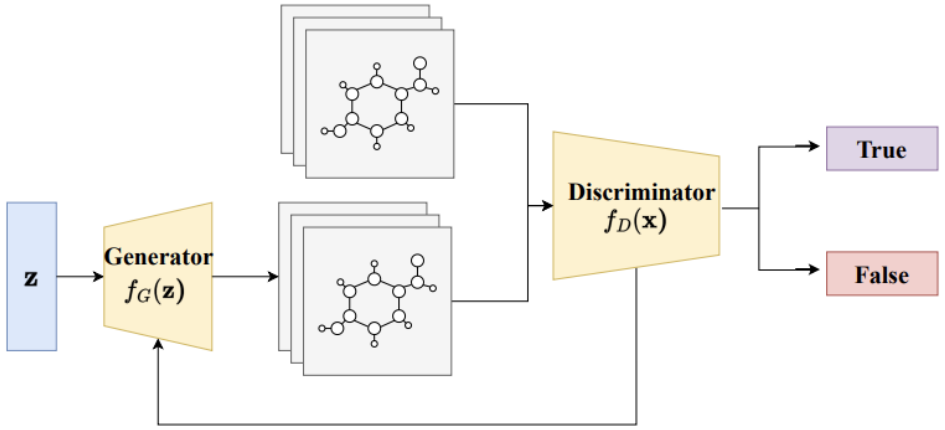
	community-small			ego-small		
Model	Degree	Cluster	Orbit	Degree	Cluster	Orbit
GraphVAE	0.35	0.98	0.54	0.13	0.17	0.05
DeepGMG	0.22	0.95	0.4	0.04	0.10	0.02
GraphRNN	0.08	0.12	0.04	0.09	0.22	0.003
GNF	0.20	0.20	0.11	0.03	0.10	0.001
GraphRNN(1024)	**0.03**	**0.01**	**0.01**	0.04	0.05	0.06
GNF(1024)	0.12	0.15	0.02	**0.01**	**0.03**	**0.0008**

https://arxiv.org/pdf/1905.13177.pdf

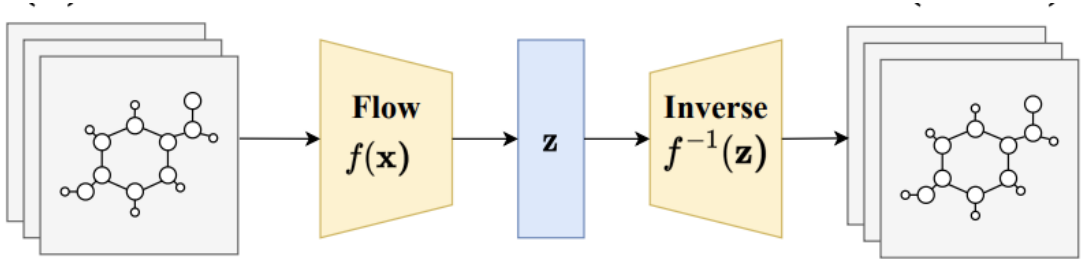
Generative Models



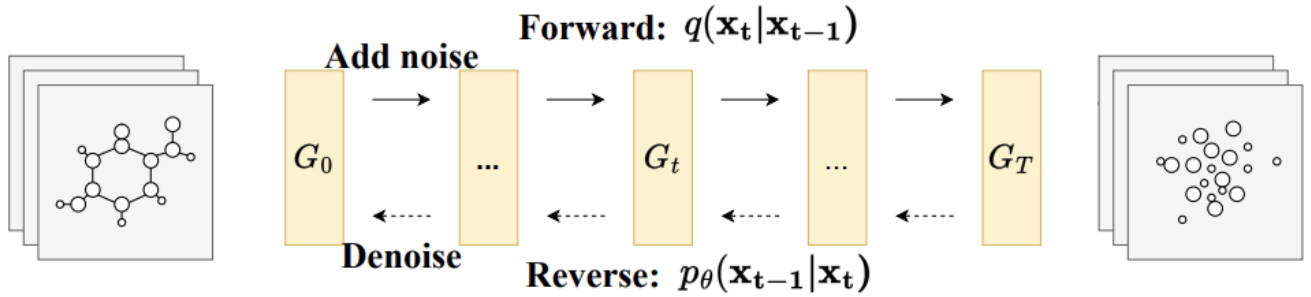
(a) Variational AutoEncoders



(b) Generative Adversarial Networks (GAN)



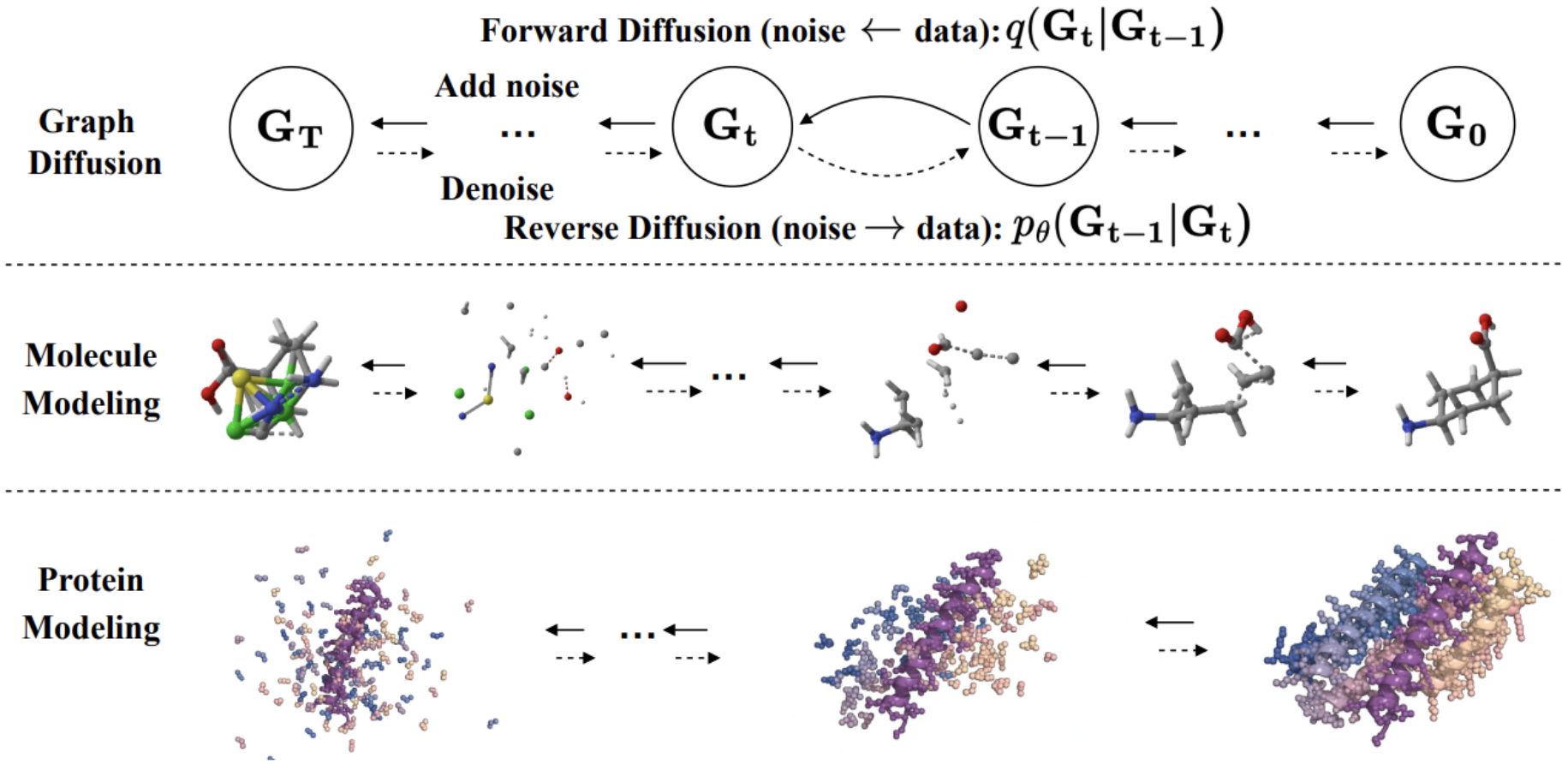
(c) Normalizing Flows



(d) Diffusion models

https://arxiv.org/pdf/2302.02591.pdf

Diffusion Models



https://arxiv.org/pdf/2302.02591.pdf

Summary

- Links are important
- Link prediction is an important self-supervised learning task
  - Learning representations
  - Learning centrality
  - Learning clustering
- Network Completion
  - More difficult task
  - Static completion
  - Dynamic completion
  - As a generative model
- Graph Generation
  - Graph RNN
  - Normalizing Flows
  - Diffusion models