Adversarial Stein Training for Graph Energy Models

Shiv Shankar
University of Massachussets
sshankar@umass.edu

Abstract

Learning distributions over graph-structured data is a challenging task with many applications in biology and chemistry. In this work we use an energy-based model (EBM) based on multi-channel graph neural networks (GNN) to learn permutation invariant unnormalized density functions on graphs. Unlike standard EBM training methods our approach is to learn the model via minimizing adversarial stein discrepancy. Samples from the model can be obtained via Langevin dynamics based MCMC. We find that this approach achieves competitive results on graph generation compared to benchmark models.

1 Introduction

Modeling and generating molecular structures is of great value in applications such as drug discovery [5] and immunology [6]. Graphs provide a useful mathematical abstraction to express and analyse molecular structures. They have also been frequently used to captures relational structure in language processing[14], systems [4] and bio-chemistry[5] . Generative models for graphs also have applications in network analysis [1], chemical analysis [20] and other fields.

Recently there have been a number of works on learning generative models of graphs from data. One framework for such models is latent variable based autoencoders. This framework includes models such as GraphVAE [31] and junction tree variational autoencoders [19]. These models typically use a graph neural networks (GNN) [12, 30] to encode the graph into a latent space, and generate samples by decoding latent variables sampled from a prior distribution. Another paradigm for graph generation is grow the graph one node (or one subgraph) at a time. Li et al. [22], You et al. [37], Liao et al. [23] take such an autoregressive approach, where graphs are generated sequentially

Our work focuses on using Energy based models (EBM) to model permutation invariant distributions on graphs. Unlike the earlier mentioned auto-encoding and autoregressive approaches, energy-based model provides an extremely flexible way to model densities. This allows one to naturally enforce desirable inductive biases and incorporate domain knowledge into the models. For example one can write models which by construction are permutation invariant by choosing appropriate energy function. EBMs and similar global unnormalized models have long been used for schema-modeling [7, 10] and structure prediction [2]. They have also shown promise in applications like speech processing [35] and protein-folding [8, 17].

In this work we utilize GNNs to construct a permutation invariant EBM for distribution over graphs. More specifically we used the learnable multi-channel approach used in [27] to design our energy function. Then we rely on using modification of Stein's Identity [33] to adversarially train EBMs without the need to sample which allows scaling to larger datasets.

2 Preliminaries

A graph G is a tuple of finite set of nodes V_G and a finite set of edges E_G which connect the nodes. Each edge e is identified by a node pair (u, v). A graph G is said to be undirected if its edges are be undirected i.e. if $(u, v) \in E_G$ then its implied that $(v, u) \in E_G$. Any unweighted graph can be represented by a symmetric matrix $\mathbf{A} \in \{0,1\}^{|V_G| \times |V_G|}$ called its adjacency matrix. We shall use the adjacency matrix representation for our graphs. In general the adjacency matrix representation $\mathbf{A}$ of a graph depends on the ordering of nodes. However in this paper we will be dealing only with permutation invariant functions. A distribution of graphs is then equivalent to a distribution of adjacency matrices $p(\mathbf{A})$.

2.1 Energy Based Models

Energy-based models (EBMs) assign to each point $\mathbf{x}$ in the input space Ω an unnormalized log probability $\mathcal{E}_\theta(\mathbf{x})$. EBMs get their name from the so-called energy function, which is simply $\mathcal{E}$. This function fully specifies the distribution over the data as :

$$p_\theta(x) = \exp(\mathcal{E}_\theta(\mathbf{x}))/Z(\theta)$$

Here, θ represents the model parameters, and $Z(\theta) = \sum_{x \in \Omega} exp(\mathcal{E}_\theta(x))$ is the normalization constant. Because $Z(\theta)$ is typically intractable, most EBMs cannot be trained by maximum likelihood and instead must rely on alternatives relying on MCMC sampling; which is unscalable for large datasets. There are also sampling-free methods to train non-normalized models. We refer to Appendix C for discussion of such methods.

2.2 Stein Discrepancy

Stein Discrepancy [11] between two distributions p and q is given by:

$$S(q,p) = \max_{f \in \mathcal{F}} \mathbb{E}_{q(z)}[(\partial_z \log p(z))^T f(z) + Tr(\partial_z f(z))]$$

where $\mathcal{F}$ is the family of function to over which the maximization is done. This was proposed as an extension Stein's Identity [33] and it can be shown that if $\mathcal{F}$ is a sufficiently rich class of functions, then the above expression is indeed a divergence [11]. Note that the terms inside the expectation can be computed even for an unnormalized distribution p since $\partial_z \log p(z)$ does not depend on the normalization constant of p.

3 Adversarial Stein Training for Graph Energy Models

In this section we present our approach for modeling graph distributions dubbed ASTRAGEM (***A***dversarial ***S***tein ***Tra***ining for ***G***raph ***E***nergy ***M***odels). Our approach consists of three components. First using an Edgewise Dense Prediction Graph Neural Network (EDPGNN) [27] architecture we write a powerful permutation invariant energy function. Second we use a novel approach (labeled Adversarial Stein Training) to learn such energy models without requiring computationally burdensome sampling during training.

We utilize the idea of estimating the Stein Discrepancy [11] via optimization over parametric function spaces. We restrict $\mathcal{F}$ to be the space of functions computable by a specific neural network architecture C which is parameterized by ψ. The discrepancy measured by a specific instance of the network C_ψ

$$S(q,p,\psi) = \mathbb{E}_{q(z)}[(\partial_z \log p(z))^T C_\psi(z) + Tr(\partial_z C_\psi(z))]$$

A quantitative value of the Stein discrepancy $\hat{S}$ can then be obtained via maximizing $S(q,p,\psi)$. From the perspective of implementation we perform this optimization via gradient descent

$$\hat{S}(q,p) = \max_{\psi} \mathbb{E}_{q(z)}[(\partial_z \log p(z))^T C_\psi(z) + Tr(\partial_z C_\psi(z))]$$

Note that computing the above discrepancy only requires samples from q. Furthermore computing the $\partial_z \log p(z)$ for an EBM is simple as:

$$\partial_z \log p(z) = \partial_z \log \exp(\mathcal{E}_\theta(z))/Z(\theta) = \partial_z \mathcal{E}_\theta(z)$$

is independent of Z and directly computable via the energy function $\mathcal{E}_\theta$. Once the discrepancy is estimated we can minimize the same over the parameters of our energy model θ. The entire procedure is analogous to training a GAN [3], hence the name adversarial training. The discrepancy estimator C_ψ can be thought of as an adversary or critic, while the generator is the energy model $\mathcal{E}_\theta$. As such we will sometimes refer to the discrepancy estimator as a "critic". For our current purpose of learning an energy model over graphs, we set q to be the empirical distribution of graphs from the data. The model probability p is chosen to be an energy model given by an EDPGNN (parameterized via θ); which makes our distribution inherently permutation invariant. Adversarially learnt models often produce realistic but non-diverse samples [29], however incorporating domain knowledge into the adversary can help alleviate such issues [36]. Our approach allows us to bring domain insights into both the generator $\mathcal{E}_\theta$ and the critic C_ψ

3.1 Noise Smoothening and Critic Regularization

Since the empirical distribution q over graphs is discrete while the p_θ is continuous the ideal critic can easily distinguish the two leading to large gradients and unstable training. In our experiments clipping or norm-regularizing the critic parameters ψ were not sufficient to alleviate the problem. As such to stabilize the training we employed multiple tricks. First in each minibatch we add Gaussian noise $\mathcal{N}(0, \sigma^2)$ to the adjacency matrices of the graphs and symmetrize them [1] before using the matrices as inputs. Secondly instead of using a single noise variance we used multiple values of σ^2 at the same time during training to get different corruptions of the same graph. Thirdly instead of a single critic we use a series of noise conditioned critics C^i_{ψ,σ_i} which are regularized by parameter sharing. Finally we add extra regularization to the critics by trying to reduce their power against a Gaussian kernel-density estimate $p_h^{KDE}(\mathbf{x})$ of the distribution. More details of the exact procedure are available in the Appendix A.

The overall training procedure is presented in Algorithm 1

Algorithm 1 Adversarial Stein Training for Graph Energy Model

```
Require: Critic architecture $C_\psi$, EBM architecture $\mathcal{E}_\theta$, data $D = \{\mathbf{A}_j\}_{j=1}^n$, noises $\{\sigma_i\}_{i=1}^k$
Require: Hyperparameters: Regularization $\lambda_K, \lambda_{L_2}$, Critic Update Steps $\mathrm{C}_{\text{iter}}$, Total Iterations $T$
    for $T$ iterations do
        Sample $\mathbf{A}^G$ from $D$
        $q_i^G = \mathbf{SYM}(\mathcal{N}(\mathbf{A}^G, \sigma_i^2))\forall i \in [1 \,..\, k]$
        for $\mathrm{C}_{\text{iter}}$ iterations do
            Sample $\mathbf{A}^C$ from $D$
            $q_i^{Cr} = \mathbf{SYM}(\mathcal{N}(\mathbf{A}^C, \sigma_i^2))\forall i \in [1 \,..\, k]$
            Update $\psi$ with $\nabla_\psi \sum_i \left(S(q_i^{Cr}, \mathcal{E}_\theta, C_{\psi,\sigma_i}) - \lambda_K |S(q_i^{Cr}, p_{\sigma_i}^{KDE}, C_{\psi,\sigma_i})|\right) - \lambda_{L_2}|\psi|^2$
        end for
        Update $\theta$ with $-\nabla_\theta \sum_i \left(S(q_i^G, \mathcal{E}_\theta, C_{\psi,\sigma_i})\right)$
    end for
    Return resulting model $\mathcal{E}_\theta$
```

3.1.1 Graph Sampling

Once we have a trained energy model, we use the following procedure to sample new graphs from the distribution. We first obtain the number of nodes N in the graph. For this the approach of Ziegler and Rush [39], Niu et al. [27] is taken which samples from the empirical multinomial distribution of node sizes in the training data. Once N is fixed, we can sample matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ via Langevin dynamics on the energy function $\mathcal{E}_\theta$.

4 Experiments

We present empirical findings regarding our training method here. The results show this method can be used to learn good-quality generative models.

[1] **SYM** refers to symmetrization operation

Model	Community-small				Ego-small			
	Deg.	Clus.	Orbit	Avg.	Deg.	Clus.	Orbit	Avg.
GraphVAE	0.350	0.980	0.540	0.623	0.130	0.170	0.050	0.117
DeepGMG	0.220	0.950	0.400	0.523	**0.040**	0.100	0.020	0.053
GraphRNN	0.080	**0.120**	0.040	0.080	0.090	0.220	0.003	0.104
ASTRAGEM	**0.050**	0.124	**0.026**	**0.067**	0.045	**0.090**	**0.006**	**0.047**

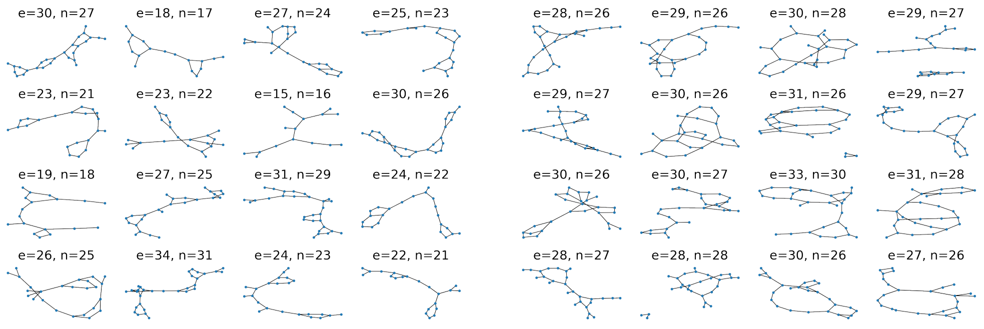
Table 1: MMD results of various graph generative models

Community Graph Learning We train EBMs on the two community graph datasets used commonly for graph learning You et al. [38] :

- Ego-small: 200 graphs with $4 \leq |V| \leq 18$, obtained from the Citeseer network.
- Community-small: 100 two-community graphs with $12 \leq |V| \leq 20$ that were generated procedurally,

We compare our approach against other recent generative models like GraphRNN [37], GraphVAE [31] and DeepGMG [22]. The evaluation method followed is the same as the one prescribed in You et al. [37] which computes MMD scores between generated samples and the test set for 3 graph statistics: degree, orbit,and cluster. Our results are summarized in Table 1 where we can see our method outperforms other methods on most metrics.

Molecule Generation The goal of molecule generation is to learn a distribution of valid molecules from a database of molecular structures. A molecule can be represented as a graph with the atoms becoming nodes, and the atomic bonds becoming edges. All the nodes and edges have associated categorical information about the type and nature of the atoms and bonds respectively. We test the performance of ASTRAGEM on the Zinc250k dataset [18].



(a) Training data (b) ASTRAGEM samples

Figure 1: Samples from the training data and ASTRAGEM on the Zinc250k dataset [18]

Our current approach only learns the graph structure and not the node and edge labels which makes comparison with existing works on the dataset difficult. As such we refrain from making any comparisons and instead present some samples as generated by our model in Figure 1.

5 Conclusion

This work explored an adversarial training approach based on Stein Divergence to learn an EBM for model graph distributions. The results show our method has potential to be a useful generative models for molecular graphs. Our method can be considered a version of adversarial inference, which opens up possibilities of bringing other advances in the field [9] into learning generative models from graphs. Furthermore, while these preliminary results seem promising, more research is needed to successfully use our model for generation of full molecular graphs i.e all atom and bond features.

References

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

[2] D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*, 2016.

[3] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[4] V. Batagelj and M. Zaversnik. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.

[5] R. Bonetta and G. Valentino. Machine learning techniques for protein function prediction. *Proteins: Structure, Function, and Bioinformatics*, 88, 10 2019.

[6] L. C. Crossman. Leveraging deep learning to simulate coronavirus spike proteins has the potential to predict future zoonotic sequences. *bioRxiv*, 2020.

[7] A. Deshpande and S. Sarawagi. Probabilistic graphical models and their role in databases. In *Proceedings of the 33rd international conference on very large data bases*, pages 1435–1436, 2007.

[8] Y. Du, J. Meier, J. Ma, R. Fergus, and A. Rives. Energy-based models for atomic-resolution protein conformations. In *International Conference on Learning Representations*, 2020.

[9] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[10] S. Godbole and S. Sarawagi. Discriminative methods for multi-labeled classification. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 22–30. Springer, 2004.

[11] J. Gorham and L. Mackey. Measuring sample quality with stein's method. In *Advances in Neural Information Processing Systems*, pages 226–234, 2015.

[12] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[13] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, and R. Zemel. Cutting out the middle-man: Training and evaluating energy-based models without sampling. *arXiv*, pages arXiv–2002, 2020.

[14] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto. Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. *arXiv preprint arXiv:1706.05674*, 2017.

[15] T. Hu, Z. Chen, H. Sun, J. Bai, M. Ye, and G. Cheng. Stein neural sampler. *arXiv preprint arXiv:1810.03545*, 2018.

[16] A. Hyvärinen. Some extensions of score matching. *Computational statistics & data analysis*, 51(5):2499–2512, 2007.

[17] J. Ingraham, A. Riesselman, C. Sander, and D. Marks. Learning protein structure with a differentiable simulator. In *International Conference on Learning Representations*, 2019.

[18] J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad, and R. G. Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7): 1757–1768, 2012.

[19] W. Jin, R. Barzilay, and T. Jaakkola. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364*, 2018.

[20] P. S. John, Y. Guan, Y. Kim, S. Kim, and R. Paton. Prediction of Homolytic Bond Dissociation Enthalpies for Organic Molecules at near Chemical Accuracy with Sub-Second Computational Cost. 11 2019.

[21] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[22] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[23] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel. Lanczosnet: Multi-scale deep graph convolutional networks. *arXiv preprint arXiv:1901.01484*, 2019.

[24] J. N. Lim, M. Yamada, B. Schölkopf, and W. Jitkrittum. Kernel stein tests for multiple model comparison. In *NeurIPS*, pages 2240–2250. 2019.

[25] Q. Liu, J. Lee, and M. Jordan. A kernelized stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pages 276–284, 2016.

[26] K. Madhawa, K. Ishiguro, K. Nakago, and M. Abe. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

[27] C. Niu, Y. Song, J. Song, S. Zhao, A. Grover, and S. Ermon. Permutation invariant graph generation via score-based generative modeling. *arXiv preprint arXiv:2003.00638*, 2020.

[28] R. Ranganath, D. Tran, J. Altosaar, and D. Blei. Operator variational inference. In *Advances in Neural Information Processing Systems*, pages 496–504, 2016.

[29] A. Risteski, Y. Zhang, and S. Arora. Do GANs learn the distribution? some theory and empirics. In *International Conference on Learning Representations*, 2018.

[30] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[31] M. Simonovsky and N. Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018.

[32] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11918–11930, 2019.

[33] C. Stein et al. A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory*. The Regents of the University of California, 1972.

[34] P. Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[35] B. Wang and Z. Ou. Learning neural trans-dimensional random field language models with noise-contrastive estimation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6134–6138. IEEE, 2018.

[36] J. Xu, X. Ren, J. Lin, and X. Sun. Dp-gan: diversity-promoting generative adversarial network for generating informative and diversified text. *arXiv preprint arXiv:1802.01345*, 2018.

[37] J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in neural information processing systems*, pages 6410–6421, 2018.

[38] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. *arXiv preprint arXiv:1802.08773*, 2018.

[39] Z. M. Ziegler and A. M. Rush. Latent normalizing flows for discrete sequences. *arXiv preprint arXiv:1901.10548*, 2019.

A Noise Smoothening and Regularization

Since the optimal critic C for a pair of distributions p, q need not be optimal for another pair, instead of a single critic using multiple critics was far more effective. We use a series of critics C_ψ^i that are conditioned on their corresponding noise levels. These are regularized by sharing parameters across the critics. For this purpose we use the following structure for the layers in the critic. Each MLP layer f_j in the noise conditional critic C_ψ^i is given as

$$f_{j,i}(\mathbf{A}) = \text{Activation}((\mathbf{W}_j\mathbf{A} + \mathbf{b}_j)\alpha_i + \beta_i)$$

where α_i, β_i are individual parameters for each noise level σ_i and $\mathbf{W}_j, \mathbf{b}_j$ are layer parameters shared across all critics. This strategy was also utilized in Niu et al. [27] to learn a family of noise conditioned score functions. Furthermore a Gaussian kernel-density estimate of the distribution is further used to provide extra regularization to the critics. Recall that for a set of samples $\mathbf{A}_{1..N}$ drawn from any distribution, its kernel density estimate is given by:

$$p_h^{KDE}(\mathbf{x}) = \sum_{n=1}^{N} K_h(\mathbf{x} - \mathbf{A}_i)$$

where K_h is the Gaussian kernel function with bandwidth h i.e $K_h \propto \exp\big(-(\frac{\mathbf{x}-\mathbf{A}_i}{h})^2\big)$. For simplicity we set the bandwidth to be the same as σ_i.

B Edgewise Dense Prediction Graph Neural Network

Niu et al. [27] devised Edgewise Dense Prediction Graph Neural Network (EDPGNN) to learn generative models for graphs. We briefly summarize their architecture here. EDPGNN extends a standard message passing model with two additions a) EDPGNN have an edge update function along with a node update function and b) EDPGNN is equipped with multiple channels akin to a convolutional network. The message passing and update operation for an EDPGNN with C channels and T message passing steps is given as:

$$\begin{aligned} m_{[c,v]}^{t+1} &= \mathbf{A}_{[c,v,w]}^{t} h_w^t \quad \forall c \in \{0, 1, ..C\} \\ h_v^{t+1} &= \text{MLP}_t^{\text{Node}}(\mathbf{CAT}[\tilde{m}_{[c,v]}^{t+1} + (1+\epsilon)m_w^t]_{c\in\{0,1,..C\}}) \\ \mathbf{A}_{[c,u,v]}^{t+1} &= \mathbf{SYM}(\text{MLP}_t^{\text{Edge}}(A_{[c,u,v]}^t, h_u^{t+1}, h_v^{t+1})) \end{aligned}$$

where **CAT** refers to concatenation operator and **SYM** refers to the symmetrization operator which symmetrizes a matrix. The basic idea is to encode different variations of the same graph in different channels, and compute messages across them. Then for prediction or node updation we aggregate information across channels via flattening the messages into a single vector.

C Related Works

Niu et al. [27] proposed a permutation invariant generative model for graphs based on DSM procedure. The architecture of the energy network in our work is the same as the one presented posed by them. The key difference between our work and [27] is two fold. Firstly instead of score matching we have an adversarial training procedure which leads to better samples. Secondly since Niu et al. [27] try to learn the score gradients directly, their model does not provide a density model and cannot be used to assess either the likelihood or unnormalized density of data samples.

[15] proposed minimizing the Stein divergence to learn an implicit sampler for a given unnormalized probability model. Their approach backpropagates the gradients of the Stein discrepancy S to the sampler via the reparametrization trick [21]. [28] introduced a similar adversarial objective for a different family of critic functions. Stein divergence based training methods for unnormalized models have also been proposed [13, 15]. Multiple works [25, 15, 24] have used the Stein divergence restricted to RKHS to build goodness of fit tests.

Multiple non-sampling methods train non-normalized models have been proposed in literature. Hyvärinen [16] proposed a score matching (SM) procedure which minimizes the Fisher divergence. Another method known as Denoising Score Matching (DSM), based on score matching with noise perturbed data was proposed by Vincent [34]. Our use of noise conditioned critics has similarities to the use of noise conditioned scores in DSM [32]. Training of graph EBMs was also recently explored [**?**]. However these models relied on MCMC based methods and were not permutation invariant.

Another class of graph generative models is based on variational flows [26]. Similar to autoregressive models, these are trained via maximum likelihood estimation estimation. While in principle these models can include constraints, in practice these works found that to be detrimental for model performance.