# Learning semi-Markovian DAGs with flow-based VAE

**Dangchan Kim, Byungguk Kang, Jaeseok Kim, Minchan Kim**

December 11, 2023

**Abstract**

We propose a method to learn the structure of a semi-Markovian directed acyclic graph, which is a mixed graph containing both directed and bidirectional edges, using a flow-based variational autoencoder. The proposed method is based on the assumption that noise variables in the linear structural equation model can be considered as latent variables. To learn the structure of the mixed graph, we employ an inverse autoregressive flow to approximate the dependency structure of the noise variables and its prior distribution. We conducted experiments on simulated data by adjusting the number of nodes and the proportion of bidirectional edges. The code is available at `https://github.io/ddangchani/NFG-VAE`.

## 1 Introduction

Directed acyclic graphs (DAGs) are commonly used to represent causal relationships between variables [1]. However, recovering the causal structure from observational data is an NP-hard problem [2]. To overcome this problem, several methods have been proposed. NOTEARS [3] is a method that learns the structure of DAGs by minimizing the continuous approximation of the discrete constraint, enabling the learning of DAGs through continuous optimization.

Based on the continuous acyclicity constraint neural network-based approaches known as graph neural networks, have been proposed. [4, 5, 6, 7] The most representative model would be DAG-GNN [4], a method that learns the structure of DAGs by using a neural network that takes the adjacency matrix of the graph as an additional input. It utilizes a variational autoencoder (VAE) to learn DAGs by interpreting the noise variable in the linear structural equation model as a latent variable.

However, in some cases, the noise variables may not be independent. For example, in a linear structural equation model with correlated hidden variables, the noise variables may exhibit a dependency structure. In such cases, edges with correlation in the noise variable are considered bidirectional edges, and these models are referred to as semi-Markovian [8]. Since the prior distribution of the noise variable is assumed to be a standard Gaussian in DAG-GNN or other graph neural networks, it becomes challenging to learn the structure of semi-Markovian DAGs. Therefore, we propose a method to learn the full covariance matrix of the noise variable using a flow-based VAE.

## 2 Preliminaries

### 2.1 Directed Acyclic Graphs and linear SEMs

Directed acyclic graphs (DAGs) are graphs that have directed edges and no cycles, which are often used for describing causality among variables. To identify causal relationships between variables, structural equation models are commonly used.

Let $A \in \mathbb{R}^{m \times m}$ be the weighted adjacency matrix of the DAG with $m$ nodes and $X \in \mathbb{R}^{m \times d}$ be a sample of a joint distribution of $m$ variables, where each row corresponds to one variable. In the literature, a variable is typically a scalar, but it can be trivially generalized to a $d$-dimensional vector under the current setting. The linear SEM model reads

$$X = A^T X + Z, \tag{1}$$

where $Z \in \mathbb{R}^{m \times d}$ is the noise matrix. When the graph nodes are sorted in the topological order, the matrix $A$ is strictly upper triangular. Hence, ancestral sampling from the DAG is equivalent to generating a random noise $Z$ followed by a triangular solve

$$X = (I - A^T)^{-1} Z, \tag{2}$$

Especially, DAG-GNN generalizes linear SEM to learn the weighted adjacency matrix $A$ of a DAG by using a deep generative model[4]. The generalized linear SEM follows

$$f_2^{-1}(X) = A^T f_2^{-1}(X) + f_1(Z), \tag{3}$$

where $f_1$ and $f_2$ are the parameterized functions that effectively perform (possibly nonlinear) transforms on $Z$ and $X$, respectively. And $f_2$ is invertible.

### 2.2 Mixed Graphs

There are several graphical representations of causal models, with DAGs being the most popular due to their simplicity. DAGs are suitable under the assumptions of causal sufficiency (i.e., no latent common causes of the observed variables), acyclicity (absence of feedback loops), and no selection bias (i.e., no implicit conditioning on a common effect of the observed variables). DAGs have many convenient properties, including the Markov property (with different equivalent formulations, the most prominent being d-separation) and a simple causal interpretation. A more general class of graphs is acyclic directed mixed graphs (ADMGs), which use bidirected edges to represent latent confounding and also have a convenient Markov property (sometimes referred to as m-separation) and causal interpretation. When the assumption of acyclicity is dropped, allowing for feedback, the more general class of directed mixed graphs (DMGs) can be used, which are naturally associated with (possibly cyclic) SEMs and can represent feedback loops. The corresponding Markov properties and causal interpretation in the cyclic case are more subtle[9].

In SEMs, if error terms are independent, we can describe the relationships between variables using a DAG. But if error terms are dependent, a SEM can be represented by a DMG which has both directed edges and bidirected edges. The former is referred to as the Markovian model. And the latter is the semi-Markovian model[8].

### 2.3 Graph Learning

#### 2.3.1 Constraint-Based and Score-Based Approaches

Most constraint-based approaches test for conditional independence in the empirical joint distribution in order to construct a graph that reflects this conditional independence. There are often multiple graphs that fulfill a given set of conditional independence, and so it is common for constraint-based approaches to output a graph representing some Markov equivalence class (MEC).

Score-based approaches test the validity of a candidate graph $\mathcal{G}$ according to some scoring function $S$. The goal is therefore stated as[10]:

$$\hat{\mathcal{G}} = \underset{\mathcal{G} \text{ over } X}{\arg\max} S(\mathcal{D}, \mathcal{G}) \tag{4}$$

where $\mathcal{D}$ represents the empirical data for variables X.

### 2.3.2 Continuous optimization-Based Approaches

Continuous optimization methods are pervasive in the field of deep learning, whereby highly parameterized networks are optimized using variations on gradient descent[11]. Recently, there have been an increasing number of methods which seek to learn structure from data, while leveraging the advantages of continuous optimization. These continuous optimization approaches recast the combinatoric graph-search problem into a continuous optimization problem[3]. In Equation 5, the left-hand side represents the traditional approach, which seeks the adjacency matrix A that minimizes some score function $S(A)$, subject to the implied graph $\mathcal{G}(A)$ being in the set of valid DAGs. The right-hand side represents a characterization of the continuous optimization problem which, again, seeks the adjacency matrix A that minimizes some score function $S(A)$, but this time subject to the constraint $h(A) = 0$. Here, $h$ is the function used to enforce acyclicity in the inferred graph.

$$\begin{aligned} \min_{A \in \mathbb{R}^{d \times d}} S(A) \qquad & \qquad \min_{A \in \mathbb{R}^{d \times d}} S(A) \\ \text{s.t.} \quad \mathcal{G}(A) \in DAGs \qquad & \qquad \text{s.t.} \quad h(A) = 0 \end{aligned} \tag{5}$$

Recently, continuous optimization-based approaches have been explored with the attention of deep learning methods. NOTEARS[3] suggests a continuous optimization algorithm to learn DAGs for the first time. DAG-GNN[4] enhances NOTEARS by incorporating neural network functions and variational inference such that the score function is the Evidence Lower BOund (ELBO). Our model extends DAG-GNN by applying normalizing flows into Variational Autoencoders.

## 3 Variational Autoencoders and Normalizing Flows

### 3.1 Variational Autoencoders

VAE(Variational Auto-Encoder)[12] is a neural network architecture that consists of a generative model and a variational inference. In VAE architecture, it is assumed that the continuous latent variable exists and its posterior probability is intractable. Therefore, variational inference is introduced to approximate the real posterior probability with ELBO(evidence lower bound).

Let $x$ be an observed variable and $z$ be a latent variable. $x$ is generated by some random process involving $z$. For parameter $\theta$, random variable $z$ is generated from prior distribution $p_\theta(z)$, then $x$ is generated from conditional distribution $p_\theta(x|z)$. And $p_\theta(x, z)$ becomes their parametric joint distribution. Given a dataset $\mathsf{X} = \{x^{(1)}, ..., x^{(N)}\}$, maximum marginal likelihood learning is to maximize log likelihoods $\log p_\theta(\mathsf{X}) = \sum_{i=1}^{N} \log p_\theta(x^{(i)})$. However, marginal likelihood $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$ is intractable in general. Instead, we introduce the model parameter $\phi$ and a parametric inference model $q_\phi(z|x)$. Then we can reformulate the original log likelihoods with $q_\phi(z|x)$.

$$\begin{aligned} \log p_\theta(x) = \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x)\right] &= \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x) + \log p_\theta(z|x) - \log p_\theta(z|x)\right] \\ &= \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x, z) - \log q_\phi(z|x)\right] + \mathbb{E}_{q_\phi(z|x)} \left[\log q_\phi(z|x) - \log p_\theta(z|x)\right] \end{aligned}$$

The second RHS term of the last equation is the KL divergence of the approximate posterior from the true posterior. Since KL divergence is non-negative, the first RHS term of the last equation can be expressed as $\mathcal{L}(\theta, \phi; x)$ and it is called variational lower bound.

$$\begin{aligned} \mathcal{L}(\theta, \phi; x) &= \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x, z) - \log q_\phi(z|x)\right] \\ \log p_\theta(x) &= \mathcal{L}(\theta, \phi; x) + D_{KL}(q_\phi(z|x) || p_\theta(z|x)) \geq \mathcal{L}(\theta, \phi; x) \end{aligned}$$

Therefore $\mathcal{L}(\theta, \phi; x)$ is the lower bound of the evidence, marginal likelihood. Under some condition of the distribution family $\{q_\phi\}$, maximizing $\mathcal{L}(\theta, \phi; x)$ with respect to $\{\theta, \phi\}$ will simultaneously maximize

log-likelihood $\log p_\theta(x)$ and minimize $D_{KL}(q_\phi(z|x)||p_\theta(z|x))$. In VAE architecture, we set $\mathcal{L}(\theta, \phi; x)$ as objective function to be maximized.

The ELBO also can be written as

$$\begin{aligned}\mathcal{L}(\theta, \phi; x) &= -\mathbb{E}_{q_\phi(z|x)}\left[\log q_\phi(z|x) - \log p_\theta(z) - \log p_\theta(x|z)\right] \\ &= -D_{KL}(q_\phi(z|x)||p_\theta(z)) + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right].\end{aligned}$$

To estimate the lower bound, sampling method is used. However, Monte Carlo estimate of the expectation over $q_\phi(z|x)$ should be differentiable with respect to $\phi$ in gradient method. Reparametrization trick is introduced to solve this problem. It is often possible to express random variable $z \sim q_\phi(z|x)$ as a deterministic variable $z = g_\phi(\epsilon, x)$ for auxiliary variable $\epsilon \sim p(\epsilon)$. This reparametrization moves the uncertainty induced by $\phi$ into $\epsilon$ and makes the whole estimation differentiable.

### 3.2 Normalizing Flows

Normalizing flows are a method of transforming a simple distribution into a complex distribution [13]. Let $z$ be a random variable with a simple distribution $p(z)$, and $x$ be a random variable with a complex distribution $p(x)$. We can transform $z_0$ into $z_T$ using a series of invertible transformations $\{f_k\}_{k=1,\ldots,T}$ as follows:

$$z_T = f_T \circ f_{T-1} \circ \cdots \circ f_1(z_0) \tag{6}$$

The probability density function of $x$ can be obtained by the change of variables formula:

$$p(z_T) = p(z_0)\left|\det\left(\frac{\partial f_T \circ f_{T-1} \circ \cdots \circ f_1(z_0)}{\partial z_0}\right)\right| \tag{7}$$

In VAE, the prior distribution of the latent variable $\mathbf{z}$ is typically assumed to be a standard Gaussian distribution or a multivariate Gaussian distribution with a diagonal covariance matrix. [12] However, to employ a more flexible prior distribution, we can utilize normalizing flows to transform the standard Gaussian distribution into a more complex distribution. This allows us to capture richer and more intricate patterns in the latent space, such as correlation between latent variables.

There are several types of normalizing flows, such as Householder flow [14], planar flow [13], etc. In this paper, we use the inverse autoregressive flow (IAF) [15], especially linear IAF. The linear IAF is defined as follows:

$$\mathbf{z_t} = \mu_\mathbf{t} + \sigma_\mathbf{t} \odot \mathbf{z_{t-1}}. \tag{8}$$

The linear IAF is the simplest case of IAF, which transforms a multivariate Gaussian with diagonal covariance to a multivariate Gaussian with full covariance using a single flow layer. The transformation is invertible and the determinant of the Jacobian is easily computable. To use linear IAF at VAE, we need to produce an extra output $\mathbf{L}(\mathbf{x})$ from the encoder network. The full-covariance Gaussian distribution is obtained by the following transformation:

$$\mathbf{z}_T = \mathbf{L}(\mathbf{x}) \cdot \mathbf{z}_0. \tag{9}$$

Note if we restrict the $\mathbf{L}(\mathbf{x})$ to be a lower triangular matrix with diagonal elements of ones, then the log-determinant of the Jacobian is zero, which is so-called volume preserving normalizing flow. [15]

With the linear IAF, the KL loss term can be written in closed form as follows:

$$\begin{aligned}D_{KL}(q_\phi(\mathbf{z}_0|\mathbf{x})\|p(\mathbf{z}_T)) &= \log q_\phi(\mathbf{z}_0|\mathbf{x}) - \log p(\mathbf{z}_T) \\ &= -\frac{1}{2}(\mathbf{z}_0 - \mu_\phi)^T\Sigma_\phi^{-1}(\mathbf{z}_0 - \mu_\phi) + \frac{1}{2}\mathbf{z}_T^T\mathbf{z}_T\end{aligned} \tag{10}$$

where $\mu_\phi$ and $\Sigma_\phi$ are the mean vector and covariance matrix obtained from the encoder network.

# 4 Method

## 4.1 Model

The overall architecture of our model is shown in Figure 1. The encoder network takes the input data matrix $\mathbf{X} \in \mathbb{R}^{B \times m \times 1}$ and adjacency matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$ as inputs and outputs the mean vector $\mu_\phi$ and (log) variance $\sigma_\phi$ of the latent variable $\mathbf{z}_0$. The decoder network takes the latent variable $\mathbf{z}_T$ and the same adjacency matrix $\mathbf{A}$ used at encoder as inputs and outputs the mean vector $\mu_\theta$ and covariance matrix $\Sigma_\theta$ of the reconstructed data matrix $\mathbf{X}$. Also, during the training process the decoder variance $\sigma_\theta$ is fixed to 1. For the size of the hidden layer, we use 64. Note that the $\mathbf{L_X}$ is an additional output from the encoder network. Since the initial output $\mathbf{L_X}$ is not lower triangular matrix, we apply the lower triangular matrix transformation to $\mathbf{L_X}$ to make it lower triangular matrix.

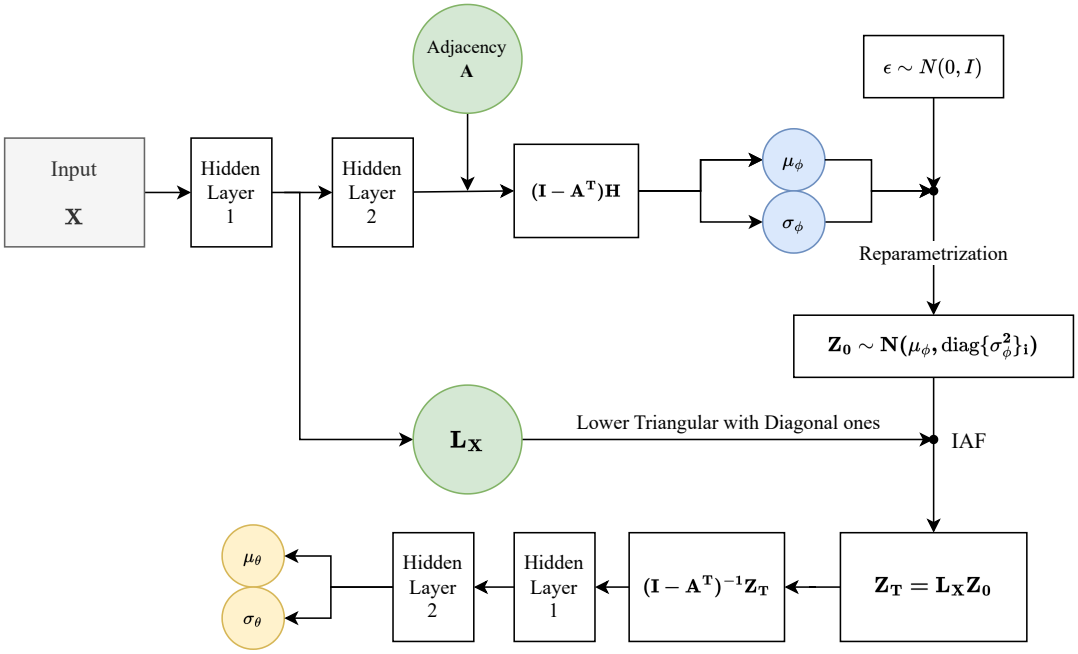



Figure 1: Architecture of the proposed model.

## 4.2 Learning

Our model aims to learn the adjacency matrix of the directed acyclic graph. Instead of regarding the mixed graph as a cyclic graph, we consider them as DAG structure with correlated noise variables. Thus we can use the same optimization procedure as DAG-GNN, which uses the acyclicity constraint as a continuous approximation of the discrete constraint. The optimization procedure is minimizing the following loss function:

$$\mathcal{L}(A, W, \lambda) = -\mathcal{L}_{\text{ELBO}} + \tau \|A\|_1 + \lambda h(A) + \frac{c}{2}|h(A)|^2. \tag{11}$$

The second term is the L1 regularization term, which encourages sparsity of the adjacency matrix. We found that $\tau = 0.1$ works well for a node size of 50. For other node sizes, we use $\tau$ proportional to

the inverse of the square of the node size. The third and fourth terms are the augmented Lagrangian terms. We gradually increase the value of $c$ during the training process, as a larger value of $c$ reduces the acyclicity constraint to zero [4].

## 5 Experiment

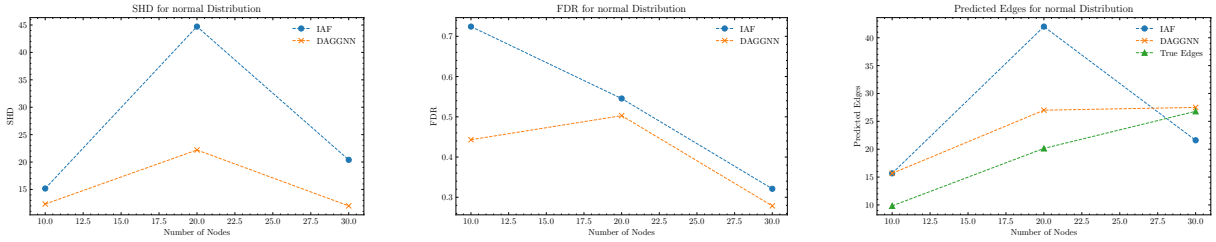
In this section, we conduct experiments on simulated data to evaluate the performance of our proposed method. We compare our method with the DAG-GNN [4] with random graph datasets. We used a thresholding value of extracting graph as 0.3 which is the same value used at DAG-GNN and NOTEARS.

**Datasets** Random graph datasets are generated using the following procedure. First, we generate a random directed acyclic graph using the Erdős–Rényi model. In Section 5.1, we consider the case where the noise variables follow not only an independent Gaussian distribution but also independent Laplace and exponential distributions. In Section 5.2, we conduct experiments where the noise variables follows a multivariate Gaussian distribution with a non-diagonal covariance matrix. We randomly select a given proportion of edges and make them bidirectional. Then, we generate a corresponding random covariance matrix and generate multivariate Gaussian data using that covariance matrix. In both sections, we generate 5000 samples for each graph. For the size of the graph, we use 10, 20, 30, and 50 nodes. For the proportion of bidirectional edges, we use 0.1, 0.3, 0.5, and 0.8.

**Evaluation** We evaluate the performance of our method using two metrics: Structural Hamming Distance (SHD) and False Discovery Rate (FDR). SHD measures the number of edge additions, deletions, and reversals required to transform the estimated graph into the true graph. FDR represents the ratio of false positives to the total number of predicted edges. These metrics are calculated by comparing the estimated graph with the true graph, considering bidirectional edges. For each combination of the number of nodes and the proportion of bidirectional edges, we generate at least 5 random graphs and calculate the average metrics.

### 5.1 Independent Noise Cases

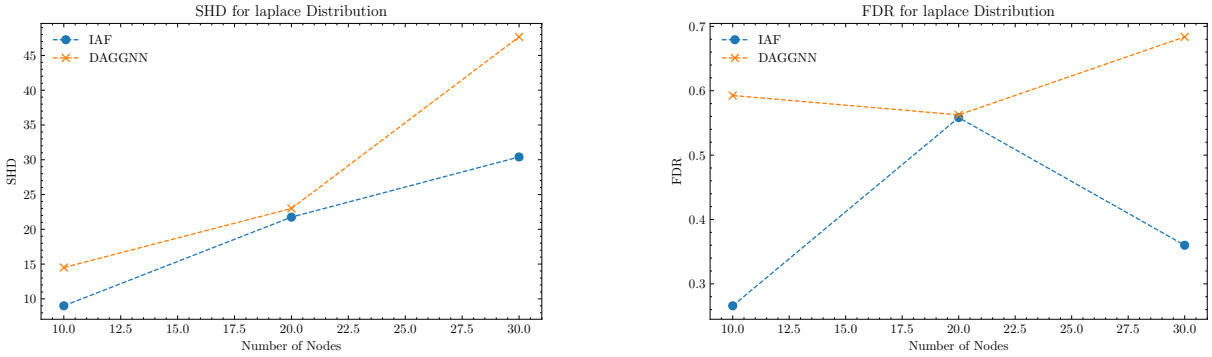
To compare the performance of our method with DAG-GNN, we conducted experiments on random graphs with independent noise variables. Specifically, we considered cases where the noise variables followed independent Gaussian, Laplace, and exponential distributions. The results of the experiments are shown in Figures 2, 3, and 4. In the case of Gaussian noise, despite using the original settings of DAG-GNN, the normalizing flow did not show superior performance compared to DAG-GNN. On average, for 10 nodes, our model and DAG-GNN achieved SHD values of 16.5 and 13, respectively. For 20 nodes, the SHD values were 24.7 and 22.1, and for 30 nodes, they were 21.8 and 18.4. However, in the case of non-Gaussian noise, our method consistently outperformed DAG-GNN in terms of both SHD and FDR as the number of nodes increased.




(a) SHD with Gaussian noise (b) FDR with Gaussian noise (c) Number of predicted edges

Figure 2: SHD, FDR, and number of predicted edges with independent Gaussian noise

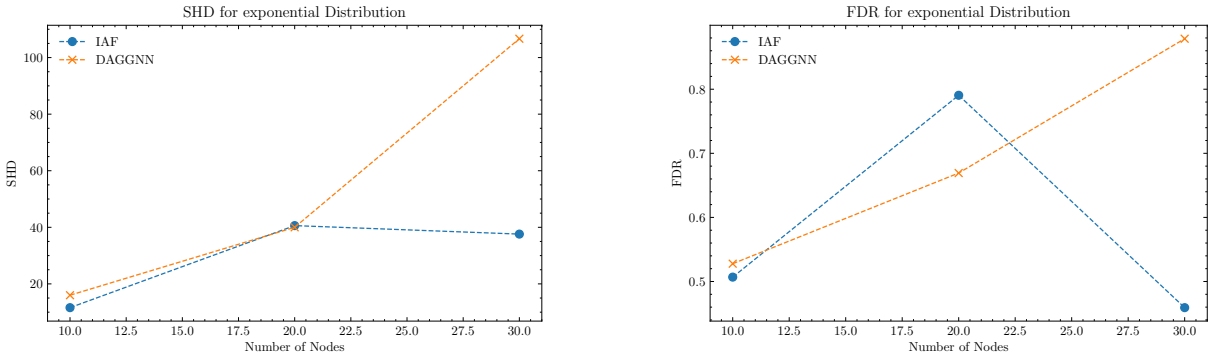
In cases where the noise variables follow Laplace or exponential distributions, our method consistently outperformed DAG-GNN in terms of both SHD and FDR. The results of the experiments are shown in Figure 3 and 4. As the number of nodes increases, our method consistently demonstrates better performance compared to DAG-GNN in terms of SHD. The FDR values of our method are also lower than those of DAG-GNN, except for the case of 20 with exponential noise.




(a) SHD with Laplace noise

(b) FDR with Laplace noise

Figure 3: SHD and FDR with independent Laplace noise

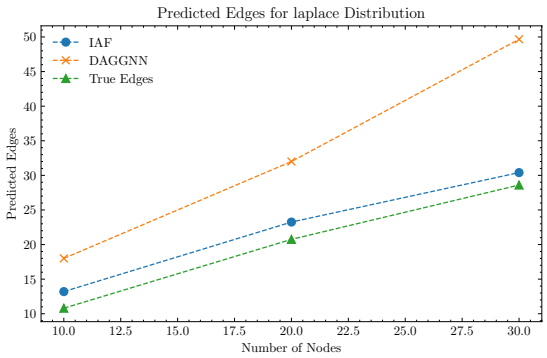



(a) SHD with exponential noise

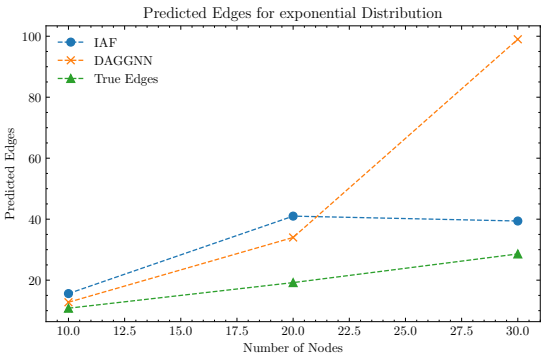
(b) FDR with exponential noise

Figure 4: SHD and FDR with independent exponential noise

Furthermore, we observed that our method produces a more accurate graph structure than DAG-GNN in cases where the noise variables are independent. As shown in Figure 5, the number of predicted edges from both methods is similar when the node size is 10. However, as the node size increases, our method consistently produces a smaller number of predicted edges compared to DAG-GNN. This is in line with our method's superior performance in terms of SHD and FDR. Figure 6 presents the estimated graphs produced by each method. The graph estimated by DAG-GNN (third column) contains considerably more edges than the true graph (first column), whereas the graph estimated by our method (second column) has a number of edges much closer to that of the true graph. Therefore, our method not only achieves better performance in terms of accuracy but also provides a more reliable estimation of the true graph structure.




(a) Predicted edges with Laplace noise




(b) Predicted edges with exponential noise

Figure 5: Number of predicted edges with Laplace and exponential noise

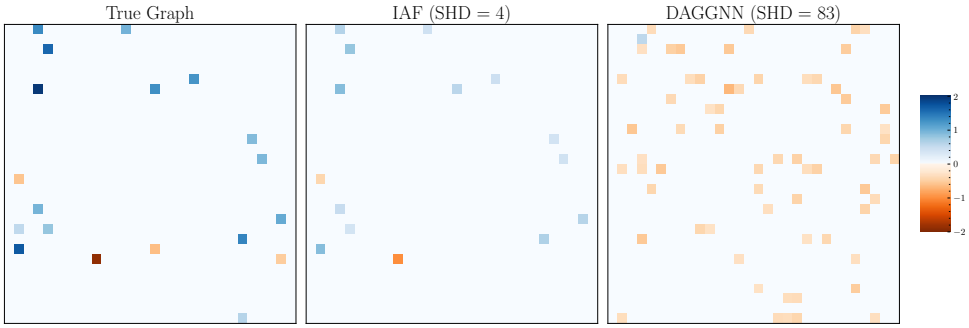
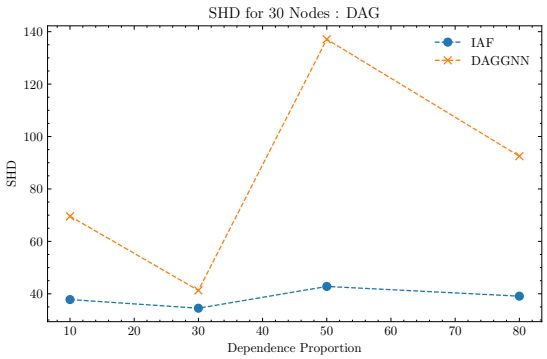


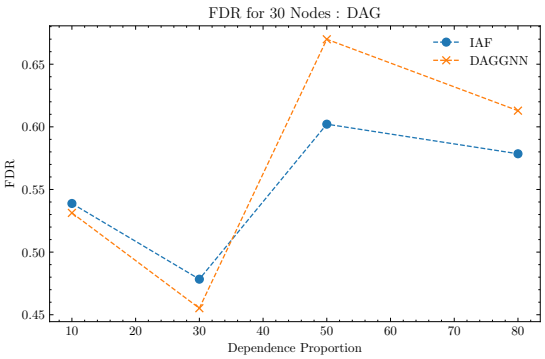

Figure 6: Example of estimated graphs with node size 30 and exponential noise

## 5.2 Dependent Noise Cases

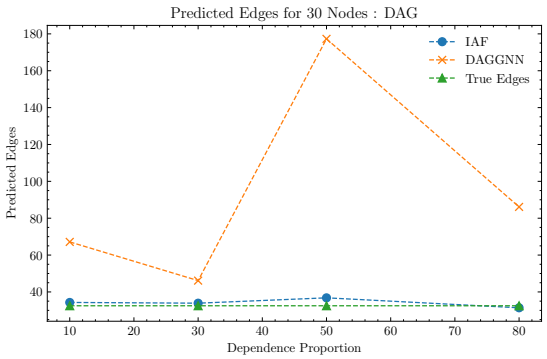
To begin with, we set the proportion of bidirectional edges to 0.3 and compared the performance of our method with that of DAG-GNN. The results, as depicted in Figure 7, demonstrate that our method outperforms DAG-GNN in terms of both SHD and FDR. Notably, with a node size of 20, our method exhibited stable training, resulting in a significant difference in SHD and FDR. Figure 8 presents the estimated graphs produced by each method. The graph estimated by DAG-GNN (third column) contains considerably more edges than the true graph (first column), whereas the graph estimated by our method (second column) has a number of edges much closer to that of the true graph.




(a) SHD with dependence proportion 0.3




(b) FDR with dependence proportion 0.3




(c) Number of predicted edges

Figure 7: SHD, FDR, and number of predicted edges with dependence proportion 0.3

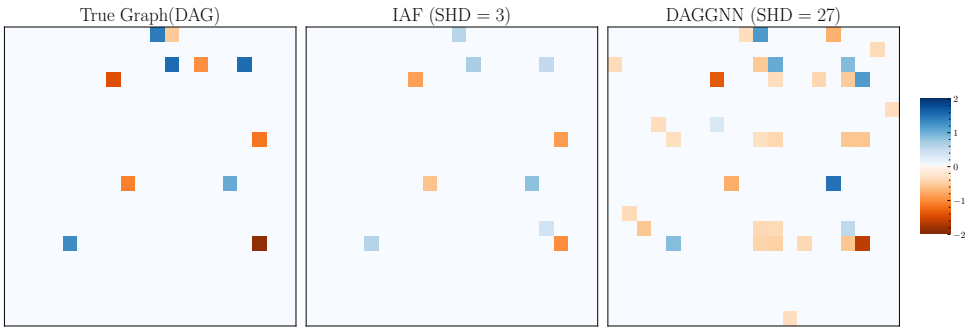
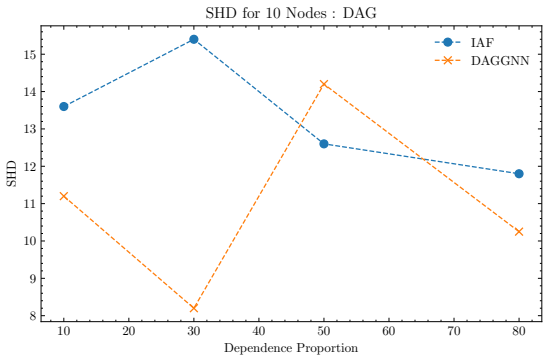


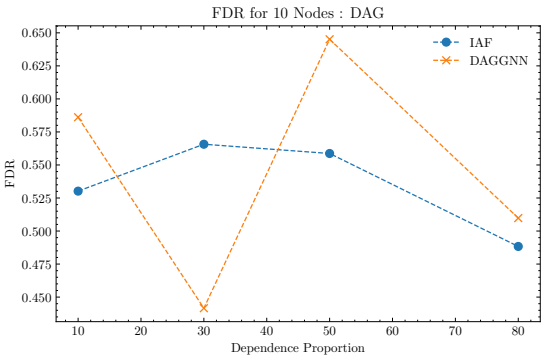

Figure 8: Example of estimated graphs with node size 20 and dependence proportion 0.5

Next, we compare the performance of the two methods, each with varying proportions of bidirectional edges. The results, as illustrated in Figure 9, reveal that when the node size is 10, DAG-GNN surpasses our method in terms of SHD. However, when the node size is increased to 20, our method takes the lead, outperforming DAG-GNN in both SHD and FDR metrics.

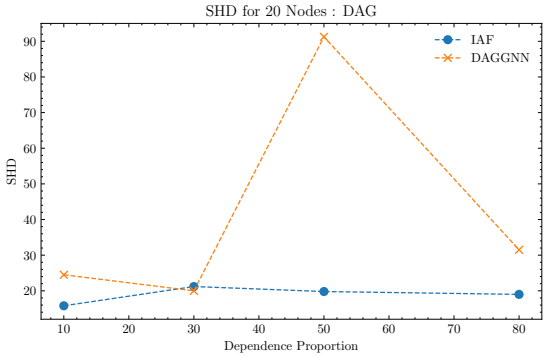
In terms of the number of predicted edges, our method consistently produces a number of edges closer to the true number compared to DAG-GNN, especially when the node size is greater than 10. As depicted in Figure 10, the number of predicted edges from both methods are similar when the node size is 10. However, for node sizes 20, 30, and 50, our method consistently produces a number of predicted edges



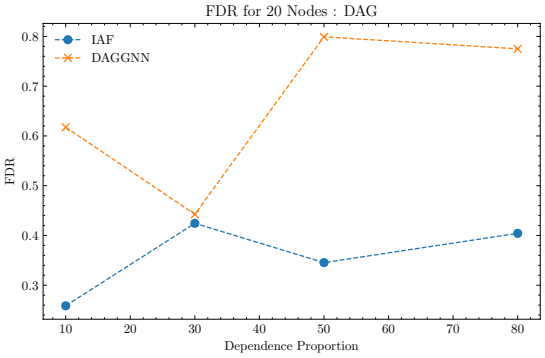

(a) SHD with node size 10



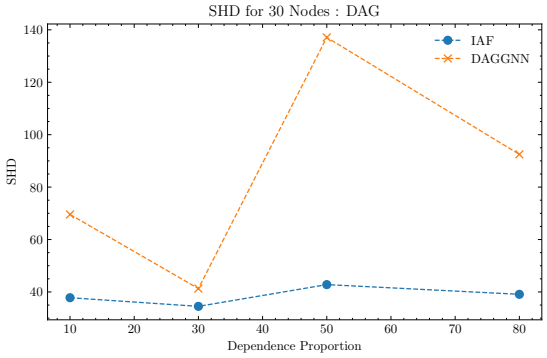

(b) FDR with node size 10



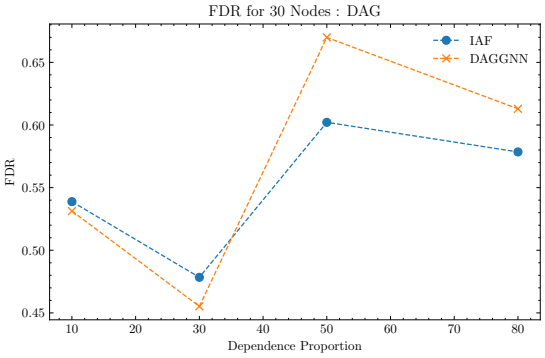

(c) SHD with node size 20



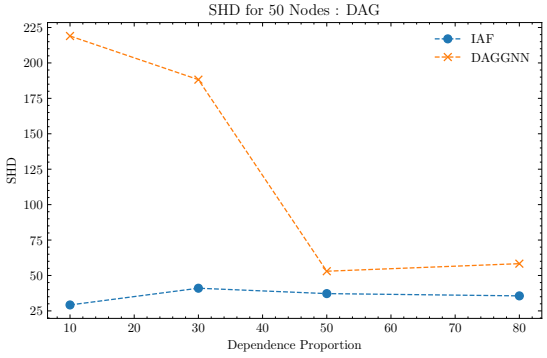

(d) FDR with node size 20



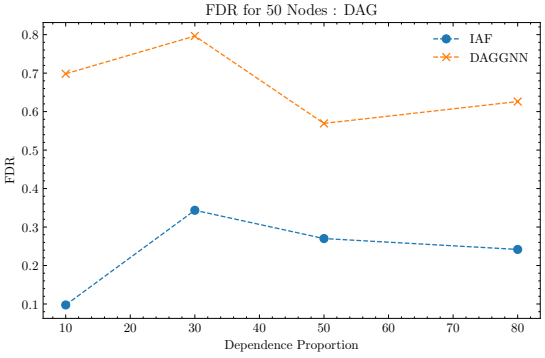

(e) SHD with node size 30




(f) FDR with node size 30

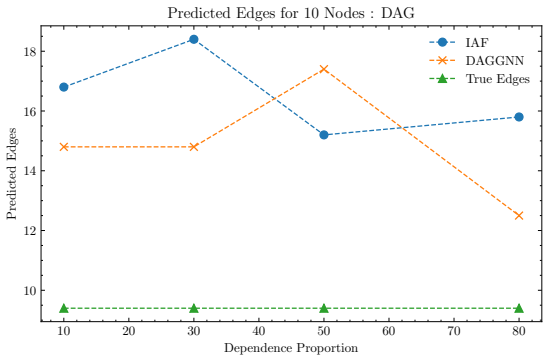



(g) SHD with node size 50

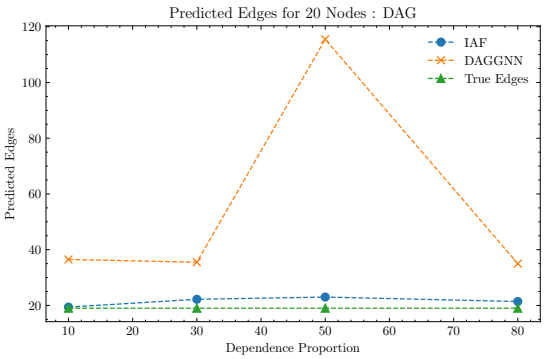



(h) FDR with node size 50

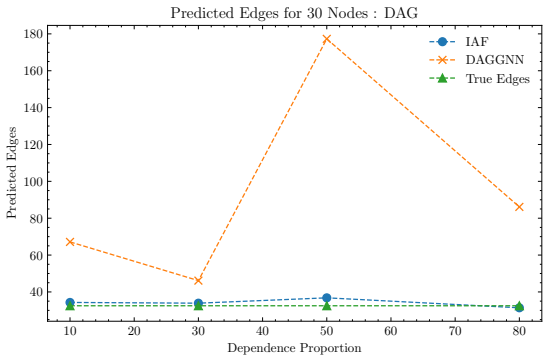
Figure 9: SHD and FDR with dependence proportion 0.1, 0.3, 0.5, and 0.8



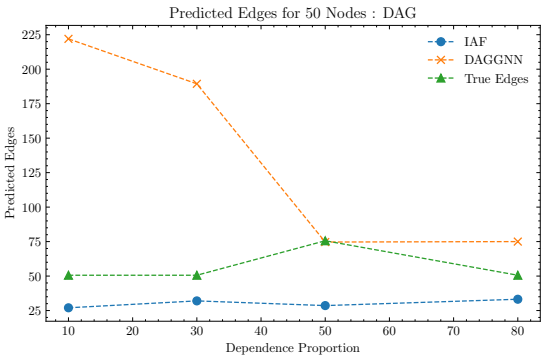

(a) Predicted edges with node size 10




(b) Predicted edges with node size 20




(c) Predicted edges with node size 30




(d) Predicted edges with node size 50

Figure 10: Number of predicted edges with dependence proportion 0.1, 0.3, 0.5, and 0.8

that aligns more closely with the true number of edges compared to DAG-GNN. Considering the better performance of our method in terms of SHD and FDR metrics, we can conclude that our method yields a more accurate graph structure than DAG-GNN.

## 6 Conclusion

In this paper, we proposed a method to learn the structure of a semi-Markovian DAG using a flow-based VAE. We conducted experiments on simulated data and compared the performance of our method with that of DAG-GNN. The results demonstrate that our method outperforms DAG-GNN in terms of both SHD and FDR metrics, especially when the noise variables have dependent structure and when the size of graph is large. In addition, our method produced a number of predicted edges closer to the true number of edges compared to DAG-GNN.

However, contrary to our initial expectations, the lower triangular matrix $\mathbf{L}$ learned in the IAF layer did not capture the covariance matrix of the actual noise variables. This seems to be due to the entanglement problem in the latent space, and resolving this issue remains a future research task. If the full covariance matrix of the noise variables could be found, it could be possible to directly identify bidirectional edges in the graph, enabling more accurate learning of the semi-Markovian graph.

## References

[1] J. Pearl, *Causality*. Cambridge University Press, 2 ed., 2009.

[2] D. M. Chickering, C. Meek, and D. Heckerman, "Large-sample learning of bayesian networks is np-hard," *CoRR*, vol. abs/1212.2468, 2012.

[3] X. Zheng, B. Aragam, P. Ravikumar, and E. P. Xing, "Dags with no tears: Continuous optimization for structure learning," 2018.

[4] Y. Yu, J. Chen, T. Gao, and M. Yu, "Dag-gnn: Dag structure learning with graph neural networks," 2019.

[5] J. Li, T. Yu, J. Li, H. Zhang, K. Zhao, Y. Rong, H. Cheng, and J. Huang, "Dirichlet graph variational autoencoder," 2020.

[6] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016.

[7] M. Zhang, S. Jiang, Z. Cui, R. Garnett, and Y. Chen, "D-vae: A variational autoencoder for directed acyclic graphs," 2019.

[8] I. Shpitser and J. Pearl, "Identification of joint interventional distributions in recursive semi-markovian causal models," in *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, p. 1219–1226, AAAI Press, 2006.

[9] S. Bongers, P. Forré, J. Peters, and J. M. Mooij, "Foundations of structural causal models with cycles and latent variables," *The Annals of Statistics*, vol. 49, Oct. 2021.

[10] J. Peters, D. Janzing, and B. Schlkopf, *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press, 2017.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. Book in preparation for MIT Press.

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

[13] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 1530–1538, PMLR, 07–09 Jul 2015.

[14] J. M. Tomczak and M. Welling, "Improving variational auto-encoders using householder flow," 2017.

[15] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved variational inference with inverse autoregressive flow," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.