NTS-NOTEARS: Learning Nonparametric DBNs With Prior Knowledge

Xiangyu Sun[1] **Oliver Schulte**[1] **Guiliang Liu**[2] **Pascal Poupart**[3]
[1]Simon Fraser University [2]The Chinese University of Hong Kong, Shenzhen [3]University of Waterloo

Abstract

We describe NTS-NOTEARS, a score-based structure learning method for time-series data to learn dynamic Bayesian networks (DBNs) that captures nonlinear, lagged (inter-slice) and instantaneous (intra-slice) relations among variables. NTS-NOTEARS utilizes 1D convolutional neural networks (CNNs) to model the dependence of child variables on their parents; 1D CNN is a neural function approximation model well-suited for sequential data. DBN-CNN structure learning is formulated as a continuous optimization problem with an acyclicity constraint, following the NOTEARS DAG learning approach (Zheng et al., 2018, 2020). We show how prior knowledge of dependencies (e.g., forbidden and required edges) can be included as additional optimization constraints. Empirical evaluation on simulated and benchmark data shows that NTS-NOTEARS achieves state-of-the-art DAG structure quality compared to both parametric and nonparametric baseline methods, with improvement in the range of 10-20% on the F1-score. We also evaluate NTS-NOTEARS on complex real-world data acquired from professional ice hockey games that contain a mixture of continuous and discrete variables. The code is available online[1].

1 INTRODUCTION

Dynamic Bayesian Networks (DBNs) are graphical models for time-series data. DBNs have many applications in real-world domains such as biology (Sachs et al., 2005), finance (Sanford and Moosa, 2012) and economics (Appiah, 2018). The paper addresses the problem of learning DBN structure from time-series data where data samples across time slices are dependent (inter-slice dependencies), and there may also exist instantaneous dependencies among variables at the same time (intra-slice dependencies).

A key issue for directed acyclic graph (DAG) learning is how to model the predictive relationship between a child node and its parents. Most previous time-series models require the user to select a priori parametric models (e.g., linear). However, when domain knowledge is not available to determine the parametric models, these approaches may lead to model misspecification and incorrect DAG structure. In this paper we develop a new approach to learn nonparametric DBNs where a child value is predicted from its parents using a 1D convolutional neural network (CNN). The 1D CNN architecture is designed to model a sequential topology in the input data, and therefore especially suitable for time-series. While a CNN defines a parameter space, it is nonparametric in the sense of being a general function approximator.

Structure Learning We formulate DBN-CNN model learning as a continuous optimization search for an acyclic weight matrix, by adapting the NOTEARS DAG learning approach for non-temporal data (Zheng et al., 2018, 2020). The weight matrix is extracted from the first layer of the trained 1D CNN kernel weights. We show analytically that using the first-layer weights involves no loss of expressive power. We show how the efficient L-BFGS-B optimization algorithm can be leveraged to incorporate useful prior knowledge in the model search, such as forbidden or required edges.

Evaluation Our evaluation focuses on apple-to-apple comparisons within the same model class as NTS-NOTEARS: temporal graphs with both intra-slice and inter-slice dependencies. Our comparison methods include representative methods based on i) nonlinear score optimization using neural networks: TCDF (Nauta et al., 2019), ii) linear models: DYNOTEARS (Pamfil et al., 2020), and iii) conditional independence (CI) constraints: PCMCI+ with nonlinear CI test (Runge, 2020).

We compare the learned structures against synthetic and real-world benchmark ground-truth DBNs (Lorenz

[1]https://github.com/xiangyu-sun-789/NTS-NOTEARS

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206. Copyright 2023 by the author(s).

Table 1: Difference between existing methods and NTS-NOTEARS. Starred methods are evaluation baselines.

Method	Score-Based	Nonlinear	Temporal	Instantaneous Edges	Acyclic
cMLP	✓	✓	✓	✗	✓
Economy-SRU	✓	✓	✓	✗	✓
GVAR	✓	✓	✓	✗	✓
VAR-LINGAM	✓	✗	✓	✓	✓
PCMCI+*	✗	✓	✓	✓	✓
TCDF*	✓	✓	✓	✓	✗
NOTEARS	✓	✗	✗	✓	✓
GraN-DAG	✓	✓	✗	✓	✓
NOTEARS-MLP	✓	✓	✗	✓	✓
DYNOTEARS*	✓	✗	✓	✓	✓
NTS-NOTEARS	✓	✓	✓	✓	✓

96 (Lorenz, 1996) and fMRI (Smith et al., 2011)), and on a new real-world dataset featuring National Hockey League (NHL) event logs. The hockey data comprise binary, categorical and continuous variables. Compared to the linear DYNOTEARS model (Pamfil et al., 2020), NTS-NOTEARS produces structures that are better, by as much as 15% in terms of F1-score on the benchmark datasets. We obtain much better improvements over the previous neural-based TCDF method (Nauta et al., 2019), due to extracting DAG edge weights from CNNs rather than the attention mechanism. Compared to the constraint-based method PCMCI+ (Runge, 2020) with nonlinear CI constraints, NTS-NOTEARS learns substantially better structures and is much more scalable.

Contributions Our contributions are as follows:

- We propose 1D CNNs to define a new class of *nonparametric DBNs* that capture linear, nonlinear, inter-slice and intra-slice relations among both continuous and discrete variables in time-series data.
- We describe NTS-NOTEARS, a continuous optimization approach for learning DBN-CNN models.
- We show how prior knowledge of dependencies can be translated into optimization constraints on convolutional weights.

The paper is structured as follows. We discuss related works in Section 2, describe NTS-NOTEARS and its training objective in Section 3 and 4, respectively. Then, we explain how to incorporate prior knowledge in Section 5. In Section 6, we evaluate NTS-NOTEARS with simulated data, benchmarks and complex real-world data.

2 RELATED WORK

Non-temporal Nonparametric DAG Structure Learning. A recent algebraic acyclicity constraint is presented in Zheng et al. (2018), as the basis of the NOTEARS approach to learn instantaneous DAGs in the linear case. Later works such as GraN-DAG (Lachapelle et al., 2019) and NOTEARS-MLP (Zheng et al., 2020) utilize the acylicity constraint for learning nonparametric nonlinear instantaneous DAGs using multilayer perceptrons (MLPs).

To understand the relationship between continuous optimization methods and previous DAG structure learning, consider a 2-stage approach: 1) For each variable X_j, learn the Markov blanket of X_j using classification/regression methods. Make each member X_i of the Markov blanket a parent of X_j. 2) Resolve cycles to produce a DAG with the same Markov blankets. Previous work using the 2-stage approach (Edera et al., 2014) proposed different discrete algorithms for each stage. Instead of 2 separate stages, the continuous optimization approach introduces 2 different components in the structure learning objective function: 1) A regression component that encourages X_i to be a parent of X_j if X_i improves the prediction of X_j. 2) An acyclicity component that discourages cycles in the resulting graph. Both predictive error and cyclicity are jointly minimized using gradient descent. If an MB discovery approach is based on a special differentiable MB predictive loss, it can be incorporated in our method by changing the loss function.

Temporal DAG Structure Learning. Learning DBNs for temporal data is a popular topic. Methods for learning DBN structure can be divided into score-based and constraint-based.

Score-based Methods. Linear autoregressive models include DYNOTEARS (Pamfil et al., 2020) and VAR-LINGAM (Hyvärinen et al., 2010). DYNOTEARS extends instantaneous linear NOTEARS using autoregression. VAR-LINGAM extends LINGAM (Shimizu et al., 2006, 2011), a linear model class with additive non-Gaussian noise. While linear models generally support fast learning, their capacity is limited compared to our nonlinear model.

There are several nonlinear neural DBN structure learning methods that estimate inter-slice dependencies only, which is consistent with Granger's approach to causality (Granger, 1969). For example, cMLP and cLSTM (Tank et al., 2021) use MLPs and LSTMs, respectively, to estimate inter-slice DBNs. GVAR (Marcinkevics and Vogt, 2021) estimates summary graphs using self-explaining neural networks. Economy-SRU (Khanna and Tan, 2019) is an RNN-based method that learns inter-slice DBNs. However, ignoring intra-slice dependencies may lead to incorrect estimation of inter-slice relations (Hyvärinen et al., 2010).

To our knowledge, TCDF (Nauta et al., 2019) is the only other method that also uses CNNs. It constructs a dependency graph structure using attention weights, rather not the NOTEARS method. The attention approach does not guarantee acylicity and is therefore a different model class from DBNs. Previous evaluations (Marcinkevics and Vogt, 2021; Khanna and Tan, 2019) and our experiments show that the attention weights do not produce accurate graphs.

Constraint-based Methods utilize CI tests to estimate graphs. PCMCI+ (Runge, 2020) outputs a completed partially directed acyclic graph (CPDAG) with multiple time steps. LPCMCI (Gerhardus and Runge, 2020) outputs a partial ancestral graph (PAG) that indicates potential latent confounders. Users can choose different CI tests based on linearity assumptions or nonparametric. However, nonlinear CI tests are computationally expensive (Zhang et al., 2011; Runge, 2018; Zheng et al., 2020; Runge et al., 2019).

Table 1 summarizes the difference between previous methods and NTS-NOTEARS. Methods excluded from our evaluation are from a different model class; we discuss them further in Appendix A.

3 NTS-NOTEARS MODEL

We work with time-series data given by $\{\boldsymbol{x}^t : t = 1, \ldots, T\}$, where $\boldsymbol{x}^t \equiv \{x_1^t, x_2^t, \ldots, x_d^t\}$ is a time-indexed d-dimensional vector of variables. For categorical variables we use one-hot encoding unless otherwise stated. A DAG G is a directed acyclic graph (V, E) where V represents vertices (i.e. nodes) and E represents edges. We assume a one-to-one correspondence between nodes and random variables and treat them interchangeably. Each edge

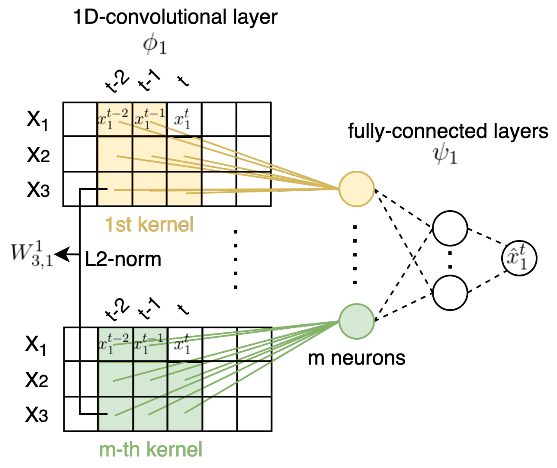


Figure 1: The architecture of NTS-NOTEARS for one child variable X_1. The convolutional weights w.r.t. the child variable in the intra-slice t are set to 0. In this example, $K = 2$ and $d = 3$. For the j-th CNN, the kernel weights are denoted by ϕ_j, the remaining parameters by ψ_j, so $\theta_j = \{\phi_j, \psi_j\}$.

$X_i \rightarrow X_j$ denotes that the variable X_j depends on the value of variable X_i. An edge $X_i^t \rightarrow X_j^t$ between variables at the same time represents an **intra-slice** or **instantaneous** dependency. An edge $X_i^{t-k} \rightarrow X_j^t$ for $k > 0$ represents an **inter-slice** or **lagged** dependency (Pamfil et al., 2020). The structure learning problem is to learn a DBN that captures the dependencies in the data $\{\boldsymbol{x}^t : t = 1, \ldots, T\}$.

Following Pamfil et al. (2020), we assume the underlying data generating process is stationary over time, and can be modelled as a K-th order Markov process, where K is a hyperparameter. These are common assumptions that can be found in related works (Runge, 2020; Khanna and Tan, 2019; Malinsky and Spirtes, 2018; Pamfil et al., 2020; Hyvärinen et al., 2010).

Temporal CNN Model Our main contribution is a new nonparametric model of acyclic temporal dependencies between the parent and child variables. We utilize 1D CNNs. A CNN exploits a sequential or grid topology in the input data. For this reason 1D CNNs are used to process sequential sensory and audio data (Yıldırım et al., 2018; Jana et al., 2020; Guan et al., 2019; Li et al., 2019; Abdoli et al., 2019). 1D CNNs learn local invariant features and aggregate them across the data sequence to learn higher-order sequence features. Higher levels in the CNN aggregate across different lags in a nonlinear trainable way.

A general MLP does not incorporate data order information. Current MLP-based methods such as cMLP (Tank et al., 2021) and IDYNO (Gao et al., 2022) concatenate the data. Data concatenation with large datasets may cause memory issues and slow down the training speed. Adding positional encoding transformer-style may be an interesting direction for future work, although the quadratic cost of self-attention is a potential concern. Appendix A contains further discussion of IDYNO.

We train d CNNs jointly where the j-th CNN predicts the expectation of the target variable X_j^t at each time step $t \geq K+1$ given preceding and instantaneous input variables:

$$\mathbb{E}[X_j^t | PA(X_j^t)] = CNN_j(\{\boldsymbol{X}^{t-k} : 1 \leq k \leq K\}, \boldsymbol{X}_{-j}^t)$$

where $PA(X_j^t)$ denotes the parents of X_j^t that are defined by the trained CNNs (see next paragraph). Here K is a hyperparameter denoting the maximum lag (order), so the input for predicting variable X_j^t comprises all preceding variables up to the maximum lag, and all variables at the same time step other than X_j. The parameters of the CNN for child variable X_j are denoted θ_j. Figure 1 illustrates our CNN architecture. The first layer of each CNN is a 1D convolution layer with m kernels, stride equal to 1 and no padding. The shape of each convolutional kernel is $d \times (K+1)$ where the last column $K+1$ represents intra-slice connections.

From Local CNNs to Model Weights. Given d parametrized CNNs, we adapt the NOTEARS-MLP approach (Zheng et al., 2020) to derive a weighted adjacency matrix W that defines a graph structure. Let $\phi_{i,j}^k \subset \theta_j$ denote the m kernel weight parameters for input variable X_i^k in the *first* convolutional layer of the j-th CNN. Each entry W_{ij}^k in the weighted adjacency matrix W represents the dependency strength of a directed edge from variable X_i^k to variable X_j^{K+1}. The estimated dependency strength of an edge between two variables is the L^2-norm of their kernel weights:

$$W_{ij}^k = ||\phi_{i,j}^k||_{L^2} \text{ for } k = 1, \ldots, K+1 \tag{1}$$

Finally, weight thresholds W_{thres}^k are applied to each time step k to prune edges with weak dependency strengths and define the parent set of each variable.

Expressive Power We next show that extracting model weights from the first CNN layer involves no loss of expressive power (cf. Zheng et al. (2020)).

Say that a function $g(X)$ is independent of input X if and only if $||\frac{\partial g(X)}{\partial X}||_{L^2} = 0$. We provide the following theorem characterizing the set of 1D CNNs that are independent of X_i^k (see Appendix B for the proof).

Theorem 1. *Let F be the class of 1D CNNs that are independent of X_i^k and F_0 be the class of 1D CNNs such that the i-th kernel parameters in the k-th column of the m first-layer CNN kernels are all zeros. Then, $F = F_0$.*

4 TRAINING OBJECTIVE

The training objective comprises four components for local functions: 1) Matching the observed child values given the parents. 2) A sparsity penalty for the CNN weights. 3) A regularization term for all parameters. 4) A cyclicity penalty to drive the induced weights to define an acyclic graph.

Let $\mathcal{L}$ denote the least-squares loss, ϕ_j^k be the concatenation of the $\phi_{i,j}^k$ vectors, and $\theta = \{\theta_1, \ldots, \theta_d\}$. The constrained training objective function is defined as:

$$\min_\theta F(\theta)$$
$$\textit{subject to } h(W^{K+1}) = 0$$

where

$$F(\theta) = \frac{1}{T-K} \cdot \sum_{t=K+1}^{T} \sum_{j=1}^{d} \mathcal{L}(X_j^t, CNN_{\theta_j}(\{\boldsymbol{X}^{t-k} : 1 \leq k \leq K\}, \boldsymbol{X}_{-j}^t)) + \sum_{k=1}^{K+1} \lambda_1^k \cdot ||\phi_j^k||_{L^1} + \frac{1}{2}\lambda_2 \cdot ||\theta_j||_{L^2}$$

$$h(W^{K+1}) = tr(e^{W^{K+1} \circ W^{K+1}}) - d = 0$$

$tr(A)$ and e^A are the trace and matrix exponential of matrix A, respectively, and $\circ$ is element-wise product. The function h enforces the acyclicity constraint among intra-slice dependencies (Zheng et al., 2020).

The augmented Lagrangian converts the constrained optimization problem to an unconstrained optimization problem. Hence, the actual (unconstrained) training objective function is:

$$\min_\theta F(\theta) + \frac{\rho}{2} \cdot (h(W^{K+1}))^2 + \alpha \cdot h(W^{K+1}) \tag{2}$$

If a variable X_i^k is predictive for the target variable X_j^t for $t \geq k$, minimizing (2) will push $\phi_{i,j}^k$ away from 0. Otherwise, the sparsity penalty and regularization will push $\phi_{i,j}^k$ towards 0. If the acyclicity constraint is violated, some parameters in ϕ^{K+1} will also be pushed towards 0 to satisfy the acyclicity constraint.

We use the L-BFGS-B algorithm (Byrd et al., 1995; Zhu et al., 1997) to optimize the unconstrained objective (2). L-BFGS (and its variants) are commonly used in works related to NOTEARS, with or without neural networks (Zheng et al., 2018, 2020; Pamfil et al., 2020; Ng et al., 2020; Yuan, 2011).

5 FROM PRIOR KNOWLEDGE TO OPTIMIZATION CONSTRAINTS

Allowing prior knowledge is often necessary for real-world applications (Shimizu et al., 2011). Adding prior knowledge about the ground-truth graph into the learning process increases not only the accuracy but also the speed of learning, since the number of parameters that need to be learned is reduced. A useful kind of prior knowledge is specifying a possible range for the dependency weights W_{ij}^k between two variables at a fixed lag. For example, specifying $W_{ij}^k = 0$ forbids an edge; specifying $W_{ij}^k \geq W_{\mathit{thres}}^k$ requires an edge (Ramsey et al., 2018). Various DBN structure learning methods make the Granger assumption that there are no intra-slice dependencies (see Section 2). By adding prior knowledge forbidding such edges, NTS-NOTEARS can leverage this assumption when valid. We show how such prior knowledge can be represented in our temporal CNN learning method, through the L-BFGS-B formulation.

According to Equation (1), the estimated dependency strength W_{ij}^k on an edge is equal to the L^2-norm of the corresponding CNN kernel parameters. Let b denote a dependency strength as prior knowledge specified by user, m be the number of kernels of the convolutional layer of each CNN, and $\bar{b}$ be the translated optimization constraints. Each b is scaled in the following way before being applied to the L-BFGS-B algorithm:

$$\bar{b} = \sqrt{\frac{b^2}{m}} \tag{3}$$

The L-BFGS-B algorithm is a second-order memory-efficient nonlinear optimization algorithm that allows bound constraints on parameters. The algorithm allows users to define which sets of parameters are free and which are constrained. For each constrained parameter, the users provide a lower bound and/or an upper bound. The bound constraints allow us to integrate prior knowledge into NTS-NOTEARS as follows.

Let $\theta = \{\dot{\theta}, \bar{\theta}\}$ where $\dot{\theta}$ denote free parameters and $\bar{\theta}$ denote constrained parameters with lower bounds l and upper bounds u, where the bounds are translated from prior knowledge according to Equation (3). The objective function (2) becomes

$$\min_{\dot{\theta}, l_1 \leq \bar{\theta}_1 \leq u_1, l_2 \leq \bar{\theta}_2 \leq u_2, \ldots} F(\theta) + \frac{\rho}{2}(h(W^{K+1}))^2 + \alpha h(W^{K+1})$$

For example, to forbid an edge from x_1 in the most recent lag to x_3, the following parameter constraint can be applied

$$\min_{\dot{\theta}, 0 \leq \phi_{1,3}^K \leq 0} F(\theta) + \frac{\rho}{2}(h(W^{K+1}))^2 + \alpha h(W^{K+1})$$

The minimum bound for NTS-NOTEARS is 0, because we take the L^2-norm of the parameters in Equation (1) and the estimated dependency strengths on edges are non-negative. Similarly, to require an edge from x_1 in the earliest lag to x_3, the following parameter constraint can be applied

$$\min_{\dot{\theta}, l \leq \phi_{1,3}^1} F(\theta) + \frac{\rho}{2}(h(W^{K+1}))^2 + \alpha h(W^{K+1})$$

where l is a positive number with $l \geq W_{\mathit{thres}}^1$.

Figure 2 illustrates the benefits of having prior knowledge. We apply NTS-NOTEARS to a simulated ground-truth DBN containing 2 time steps and 7 nodes per time step. Please see the caption for a detailed explanation. It shows that providing prior knowledge via optimization constraints may help to recover edges that are not explicitly encoded by the prior knowledge.

6 EVALUATION

All the experiments were performed on a computer equipped with an Intel Core i7-6850K CPU at 3.60GHz, 32GB memory and an Nvidia GeForce GTX 1080Ti GPU. All datasets are normalized to have mean 0 and standard deviation 1 to remove patterns in marginal variance (Reisach et al., 2021).

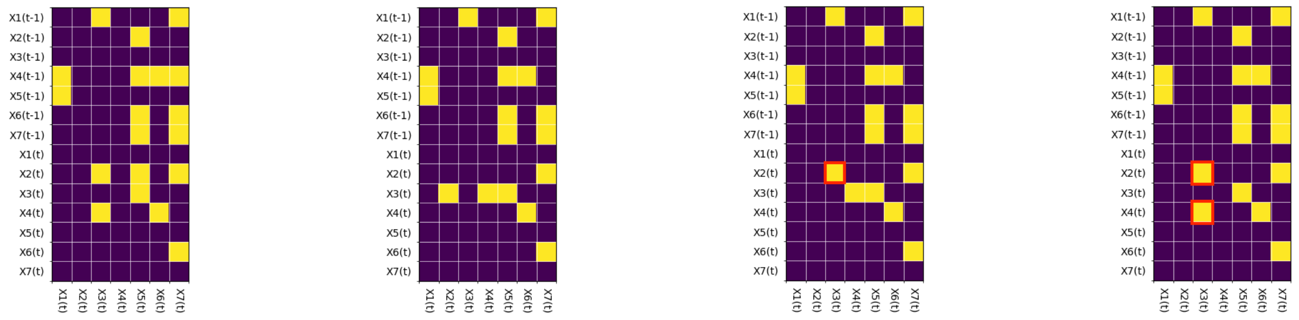
Comparison Methods We compare NTS-NOTEARS with several recent structure learning methods: TCDF (Nauta et al., 2019), DYNOTEARS (Pamfil et al., 2020) and PCMCI+ with GPDC nonlinear CI test (Runge, 2020). Note that PCMCI+ outputs a CPDAG with undirected edges. We evaluate it favourably by counting the undirected edges as correctly oriented regardless of the ground-truth edge direction. We follow the closely related DYNOTEARS work and report *F1-scores as our main metric for comparing learned graphs to ground-truth graphs.* Results for other metrics (SHD, precision and recall) are reported in the appendix.

6.1 Simulated Data

We generate 48 synthetic parametrized DBN models and then evaluate the DBN structure learned against data sampled from each ground-truth model. For generating synthetic DAGs, we follow Pamfil et al. (2020). For sampling from a model, we extend the simulator[2] provided by NOTEARS (Zheng et al., 2020, 2018) to temporal data. Figure 5 in the appendix shows how the simulated data is generated.

DBN Generation. The random ground-truth DAGs are generated based on either the Erdos-Renyi (ER) scheme (Newman, 2018) or the Barabasi-Albert (BA) scheme (Barabási and Albert, 1999) by varying the number of nodes $(K +$

[2]https://github.com/xunzheng/notears



(a) Ground-truth DBN

(b) No prior knowledge

(c) Require $X_2^t \to X_3^t$.

(d) Forbid $X_3^t \to X_2^t$

Figure 2: Each row represents a parent variable and each column represents a child variable. For instance, the yellow top-right cell represents the edge $X_1^{t-1} \to X_7^t$. Figures 2b–2d show DBNs learned by NTS-NOTEARS with various types of prior knowledge. (2a) The random ground-truth DBN. (2b) Without prior knowledge, the method recovers most of the true edges except $X_2^t \to X_3^t$ (reversed), $X_4^t \to X_3^t$ (reversed), $X_2^t \to X_5^t$ (missed), and $X_4^{t-1} \to X_7^t$ (missed). The structural Hamming distance (SHD) between the ground-truth graph and the estimated graph is 4. (2c) By simply adding a lower bound constraint $W_{thres}^2 \leq W_{2,3}^2$ that requires the edge $X_2^t \to X_3^t$, the method recovers the true edge $X_2^t \to X_3^t$. The SHD is reduced to 3. (2d) With the edge $X_3^t \to X_2^t$ forbidden, the method recovers not only the true edge $X_2^t \to X_3^t$ that is directly relevant to the provided prior knowledge but also another true edge $X_4^t \to X_3^t$ that is not directly relevant. With this prior knowledge alone, the SHD is reduced to 2.

$1) \times d$ and mean out-degrees. Given a random ground-truth DAG, the data is simulated based on one of the following three identifiable nonlinear structural causal models (SCMs): additive noise models (ANM) (Peters et al., 2017), additive index models (AIM) (Yuan, 2011; Alquier and Biau, 2013) and generalized linear models with Poisson distribution (GLM-Pois) (Park and Park, 2019). The data simulated with GLM-Pois is discrete, the data simulated by the other models is continuous with Gaussian noise. The SCM parameters are generated by uniform sampling from a closed interval following Zheng et al. (2020); Pamfil et al. (2020). The number of lags is set to 3 in the ground-truth models. Please see Appendix C for more simulation details.

Data Generation. For each ground-truth DBN, we generate training data with two sequence lengths $T \in \{200, 1000\}$. We create validation data sets as follows. For graph sizes $\{20, 40, 60, 80\}$, reference DBNs are generated using BA, AIM, intra-slice mean out-degree equal to 2, and inter-slice mean out-degree equal to 1. Then sample from each reference DBN two sequences, one for each length $T \in \{200, 1000\}$, with number of lags $= 3$.

Hyperparameters For each method and each sequence length, we select hyperparameters with a grid search that maximizes the F1-score, averaged over the validation sets for each sequence length. These hyperparameters are used as defaults for all synthetic datasets. Performance can be further improved by tuning hyperparameters to each dataset through cross-validation. However, using default hyperparameters supports assessing the general approach, as noted by Ng et al. (2020); Zheng et al. (2020). Please see Appendix D for hyperparameter values.

Results Figure 4 shows the F1-score depending on sequence length T, SCM, mean out-degrees and graph size. All methods pass the sanity check of improving with more data. *NTS-NOTEARS achieves the highest F1-scores in 20 out of 24 settings.* The score of NTS-NOTEARS is much better than that of TCDF, which shows the strength of the NOTEARS approach of defining edge weights over the attention-based approach used by TCDF. On other metrics (e.g., SHD), we also find that NTS-NOTEARS scores the best; please see Appendix E for detailed results.

6.1.1 Running Time

Figure 3 compares the average running time of the evaluation methods over 10 datasets. The neural networks and the GPDC CI test are accelerated by the same GPU. The linear method DYNOTEARS is the fastest. The constraint-based method PCMCI+ with the nonlinear GPDC CI test is substantially slower than the neural-based methods such as TCDF and NTS-NOTEARS. NTS-NOTEARS therefore offers a sweet spot trade-off between speed and learning performance.

6.2 Benchmark Data: Lorenz 96 & fMRI

Lorenz 96 (Lorenz, 1996) and fMRI (Smith et al., 2011) are two common benchmarks to evaluate causal discovery algorithms with nonlinear time-series data and nonlinear Granger causality algorithms (Nauta et al., 2019; Monti

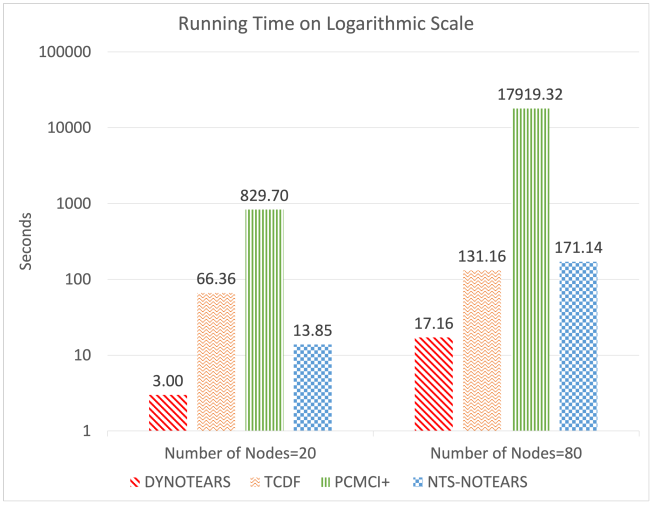


Figure 3: The average running time over 10 datasets measured in seconds with additive noise model, $K = 3$, $T = 1000$ and ER(2,1). The heights of bars are on a logarithmic scale.

et al., 2020; Marcinkevics and Vogt, 2021; Tank et al., 2021; Khanna and Tan, 2019). The Lorenz 96 model is popular in climate science as a testbed for chaotic behaviors (Schneider et al., 2017). The data follows the nonlinear dynamics given by:

$$\frac{dx_i^{t+1}}{dt} = (x_{i+1}^t - x_{i-2}^t) \cdot x_{i-1}^t - x_i^t + F$$

where F controls the chaoticity of the system. Similar to previous work (Marcinkevics and Vogt, 2021; Khanna and Tan, 2019), we consider two settings where $F \in \{10, 40\}$. The fMRI benchmark contains rich, realistic simulated blood-oxygen-level-dependent time-series for modelling brain networks. Each node in the network represents a region of interest in the brain. The ground-truth DAG in each benchmark has 2 time steps (see Figure 9 in the appendix).

In Table 2, we evaluate the methods and report their mean F1-scores and standard errors (SE) with 5 datasets sampled from the Lorenz 96 benchmark where each dataset has $d = 20$ and $T = 500$, and 10 datasets sampled from the fMRI benchmark where each dataset has $d = 5$ and $T \in \{200, 1200, 5000\}$. Please see Appendix F for hyperparameter values. *NTS-NOTEARS achieves the best F1-scores with both benchmarks, by a margin of more than 10%.* NTS-NOTEARS also achieves the best scores for other metrics (please see Appendix G).

Similar to Marcinkevics and Vogt (2021); Khanna and Tan (2019), in Table 3 we also report the area under the receiver operating characteristic curve (AUROC) by varying the hyperparameters of each method. *NTS-NOTEARS achieves the best AUROC with both benchmarks.*

Table 2: **Mean F1-scores ($\pm$ SE)** computed with Lorenz 96 and fMRI benchmarks.

Method	Lorenz 96	fMRI
DYNOTEARS	0.855 ($\pm$ 0.016)	0.475 ($\pm$ 0.020)
TCDF	0.459 ($\pm$ 0.017)	0.347 ($\pm$ 0.059)
PCMCI+	0.637 ($\pm$ 0.028)	0.502 ($\pm$ 0.045)
NTS-NOTEARS	**0.996** ($\pm$ **0.002**)	**0.628** ($\pm$ **0.023**)

Table 3: **AUROC** computed with Lorenz 96 and fMRI benchmarks by varying the hyperparameters of each evaluation method.

Method	Lorenz 96	fMRI
DYNOTEARS	0.788	0.708
TCDF	0.585	0.612
PCMCI+	0.706	0.743
NTS-NOTEARS	**0.811**	**0.749**

6.3 Real-World Ice Hockey Data

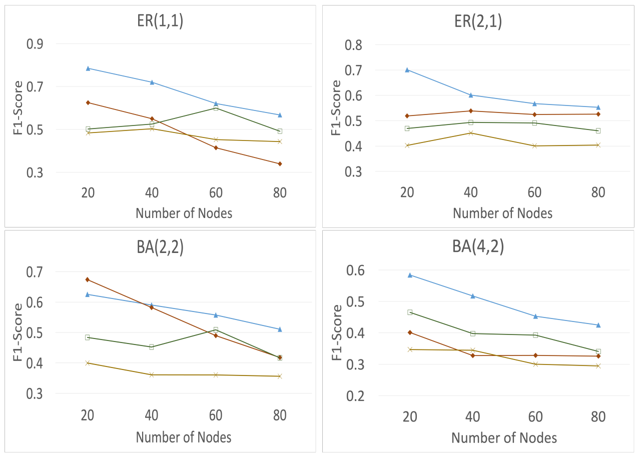
We apply NTS-NOTEARS to real-world data collected by Sportlogiq from ice hockey games in the 2018-2019 NHL season. The dataset contains a mixture of continuous, binary and categorical variables. The ground-truth distribution of each variable is unspecified. Please see Appendix H for data description. Since the play restarts after a goal is scored (i.e. face-off), we incorporate the prior knowledge that forbids edges coming from *goal(t)* or *goal(t-1)*. Because DYNOTEARS does not provide a way to incorporate prior knowledge, we manually remove any outgoing edges coming from *goal(t)* or *goal(t-1)*. We set the DYNOTEARS hyperparameters so that both methods produce the same number of edges for comparability (see Appendix H). The estimated DBNs capture many meaningful relationships between variables. An interesting question to ask in ice hockey is "what contributes to a goal?" (Sun et al., 2020; Schulte et al., 2017). By identifying the parent nodes of *goal(t)* in the DBN estimated by NTS-NOTEARS, we can answer the question: the preceding shot, the duration of the shot, the distance between the shot and the net (i.e. *xAdjCoord(t-1)*), the manpower situation and the velocity of the puck are important for scoring a goal. However, due to nonlinearity, DYNOTEARS fails to identify several goal contributors such as the duration of the shot, the distance between the shot and the net (i.e. *xAdjCoord(t-1)*), and the manpower situation. NTS-NOTEARS captures them all. Please see Figure 11 in the appendix for the learned DBNs.

7 CONCLUSION

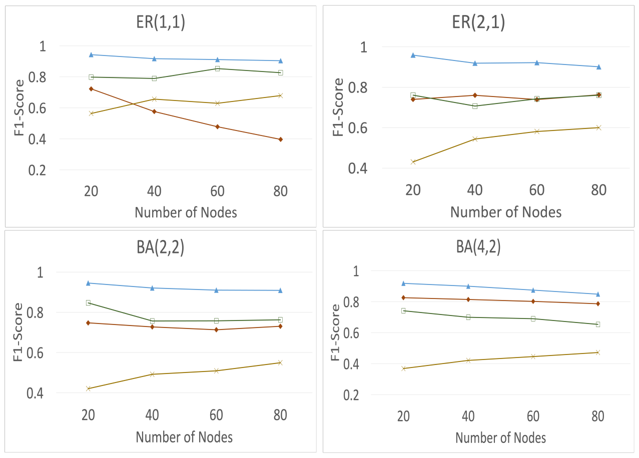
This paper described NTS-NOTEARS for learning non-parametric DBNs, a score-based structure learning method using 1D CNNs for time-series data, either with or without prior knowledge of dependencies. The learned DBNs capture both inter-slice and intra-slice dependencies. The system is user-friendly in that it supports both continuous and discrete data, and does not require knowledge of independence tests or parametric data generation models. We showed how to adapt the NOTEARS continuous optimization strategy (Zheng et al., 2018) for 1D CNNs, which allows us to learn intra-slice edges with an acyclicity constraint. Based on simulated data and standard benchmarks, we show the superior DBN structure learning quality and running speed of NTS-NOTEARS compared to several comparison methods, and demonstrate the advantage of providing prior knowledge using optimization constraints. We also apply the NTS-NOTEARS to a complex real-world sports dataset that contains a mixture of continuous and discrete variables without knowing the ground-truth underlying data distribution. A next step for future work is to extend NTS-NOTEARS to causal modelling, in particular to causal graph learning in the presence of latent confounders.

Acknowledgements

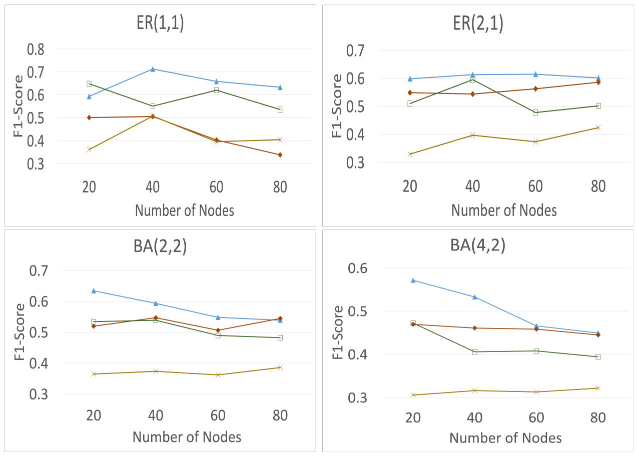
This work was supported by a Discovery Grant for Oliver Schulte from the Natural Sciences and Engineering Research Council of Canada.



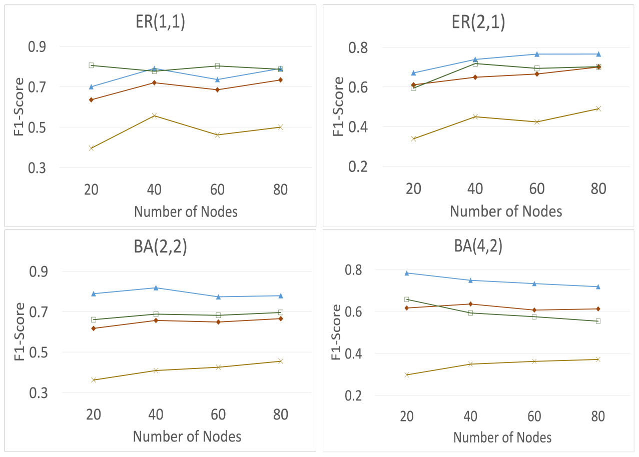
(a) Additive Index Model, $T = 200$



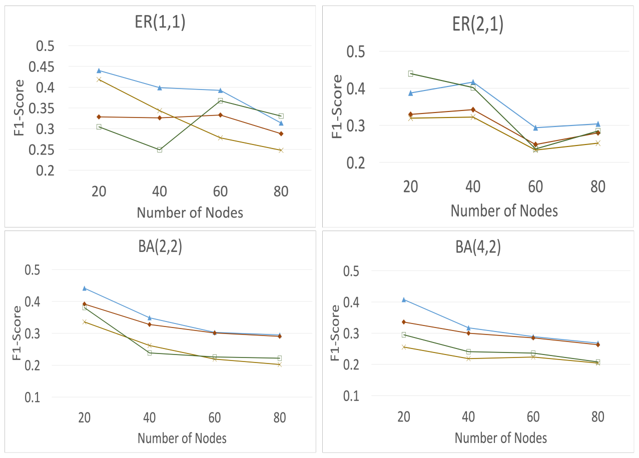
(b) Additive Index Model, $T = 1000$



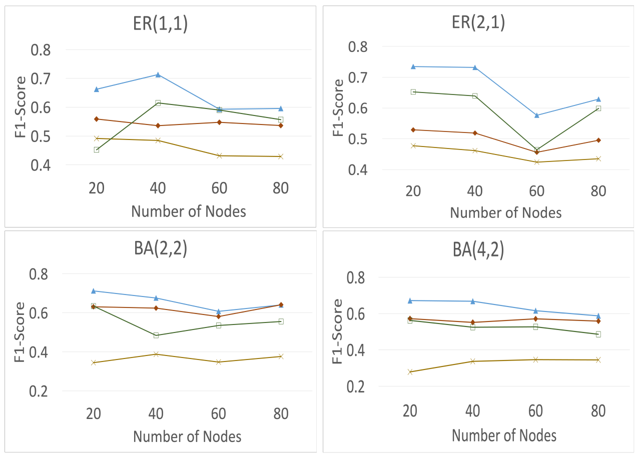
(c) Additive Noise Model, $T = 200$



(d) Additive Noise Model, $T = 1000$



(e) Generalized Linear Model with Poisson Distribution, $T = 200$



(f) Generalized Linear Model with Poisson Distribution, $T = 1000$

Figure 4: **Mean F1-scores** over 10 datasets for each setting with simulated data. Higher F1-score is better. The number of lags = 3. ER(2,1) denotes that the ground-truth DAGs are sampled using ER scheme with an intra-slice mean out-degree equal to 2 and inter-slice mean out-degree equal to 1. NTS-NOTEARS achieves the highest F1-scores in the vast majority of the settings.

References

Abdoli, S., Cardinal, P., and Koerich, A. L. (2019). End-to-end environmental sound classification using a 1D convolutional neural network. *Expert Systems with Applications*, 136:252–263.

Alquier, P. and Biau, G. (2013). Sparse single-index model. *Journal of Machine Learning Research*, 14(1).

Appiah, M. O. (2018). Investigating the multivariate Granger causality between energy consumption, economic growth and CO2 emissions in Ghana. *Energy Policy*, 112:198–208.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *science*, 286(5439):509–512.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208.

Edera, A., Strappa, Y., and Bromberg, F. (2014). The grow-shrink strategy for learning Markov network structures constrained by context-specific independences. In *Ibero-American Conference on Artificial Intelligence*, pages 283–294. Springer.

Gao, T., Bhattacharjya, D., Nelson, E., Liu, M., and Yu, Y. (2022). IDYNO: Learning nonparametric DAGs from interventional dynamic data. In *International Conference on Machine Learning*, pages 6988–7001. PMLR.

Gerhardus, A. and Runge, J. (2020). High-recall causal discovery for autocorrelated time series with latent confounders. *Advances in Neural Information Processing Systems*, 33:12615–12625.

Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438.

Guan, L., Hu, F., Al-Turjman, F., Khan, M. B., and Yang, X. (2019). A non-contact paraparesis detection technique based on 1D-CNN. *IEEE Access*, 7:182280–182288.

Hyvärinen, A., Zhang, K., Shimizu, S., and Hoyer, P. O. (2010). Estimation of a structural vector autoregression model using non-Gaussianity. *Journal of Machine Learning Research*, 11(5).

Jana, G. C., Sharma, R., and Agrawal, A. (2020). A 1D-CNN-spectrogram based approach for seizure detection from EEG signal. *Procedia Computer Science*, 167:403–412.

Khanna, S. and Tan, V. Y. (2019). Economy statistical recurrent units for inferring nonlinear Granger causality. In *International Conference on Learning Representations*.

Lachapelle, S., Brouillard, P., Deleu, T., and Lacoste-Julien, S. (2019). Gradient-based neural DAG learning. In *International Conference on Learning Representations*.

Li, Y., Baidoo, C., Cai, T., and Kusi, G. A. (2019). Speech emotion recognition using 1D CNN with no attention. In *2019 23rd international computer science and engineering conference (ICSEC)*, pages 351–356. IEEE.

Lorenz, E. N. (1996). Predictability: A problem partly solved. In *Proc. Seminar on predictability*, volume 1.

Malinsky, D. and Spirtes, P. (2018). Causal structure learning from multivariate time series in settings with unmeasured confounding. In *Proceedings of 2018 ACM SIGKDD workshop on causal discovery*, pages 23–47. PMLR.

Marcinkevics, R. and Vogt, J. E. (2021). Interpretable models for Granger causality using self-explaining neural networks. In *9th International Conference on Learning Representations (ICLR 2021)*.

Monti, R. P., Zhang, K., and Hyvärinen, A. (2020). Causal discovery with general non-linear relationships using non-linear ICA. In *Uncertainty in Artificial Intelligence*, pages 186–195. PMLR.

Nauta, M., Bucur, D., and Seifert, C. (2019). Causal discovery with attention-based convolutional neural networks. *Machine Learning and Knowledge Extraction*, 1(1):312–340.

Newman, M. E. (2018). Network structure from rich but noisy data. *Nature Physics*, 14(6):542–545.

Ng, I., Ghassami, A., and Zhang, K. (2020). On the role of sparsity and DAG constraints for learning linear DAGs. *Advances in Neural Information Processing Systems*, 33:17943–17954.

Pamfil, R., Sriwattanaworachai, N., Desai, S., Pilgerstorfer, P., Georgatzis, K., Beaumont, P., and Aragam, B. (2020). DYNOTEARS: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics*, pages 1595–1605. PMLR.

Park, G. and Park, S. (2019). High-dimensional Poisson structural equation model learning via \ell_1-regularized regression. *J. Mach. Learn. Res.*, 20:95–1.

Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of causal inference: foundations and learning algorithms*. The MIT Press.

Ramsey, J. D., Zhang, K., Glymour, M., Romero, R. S., Huang, B., Ebert-Uphoff, I., Samarasinghe, S., Barnes, E. A., and Glymour, C. (2018). TETRAD—A toolbox for causal discovery. In *8th International Workshop on Climate Informatics*.

Reisach, A., Seiler, C., and Weichwald, S. (2021). Beware of the simulated DAG! causal discovery benchmarks may be easy to game. *Advances in Neural Information Processing Systems*, 34.

Runge, J. (2018). Conditional independence testing based on a nearest-neighbor estimator of conditional mutual information. In *International Conference on Artificial Intelligence and Statistics*, pages 938–947. PMLR.

Runge, J. (2020). Discovering contemporaneous and lagged causal relations in autocorrelated nonlinear time series datasets. In *Conference on Uncertainty in Artificial Intelligence*, pages 1388–1397. PMLR.

Runge, J., Nowack, P., Kretschmer, M., Flaxman, S., and Sejdinovic, D. (2019). Detecting and quantifying causal associations in large nonlinear time series datasets. *Science Advances*, 5(11):eaau4996.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., and Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529.

Sanford, A. D. and Moosa, I. A. (2012). A Bayesian network structure for operational risk modelling in structured finance operations. *Journal of the Operational Research Society*, 63(4):431–444.

Schneider, T., Lan, S., Stuart, A., and Teixeira, J. (2017). Earth system modeling 2.0: A blueprint for models that learn from observations and targeted high-resolution simulations. *Geophysical Research Letters*, 44(24):12–396.

Schulte, O., Zhao, Z., Javan, M., and Desaulniers, P. (2017). Apples-to-apples: Clustering and ranking NHL players using location information and scoring impact. In *Proceedings of the MIT Sloan Sports Analytics Conference*.

Shimizu, S., Hoyer, P. O., Hyvärinen, A., Kerminen, A., and Jordan, M. (2006). A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(10).

Shimizu, S., Inazumi, T., Sogawa, Y., Hyvärinen, A., Kawahara, Y., Washio, T., Hoyer, P. O., and Bollen, K. (2011). DirectLiNGAM: A direct method for learning a linear non-Gaussian structural equation model. *The Journal of Machine Learning Research*, 12:1225–1248.

Smith, S. M., Miller, K. L., Salimi-Khorshidi, G., Webster, M., Beckmann, C. F., Nichols, T. E., Ramsey, J. D., and Woolrich, M. W. (2011). Network modelling methods for FMRI. *Neuroimage*, 54(2):875–891.

Sun, X., Davis, J., Schulte, O., and Liu, G. (2020). Cracking the black box: Distilling deep sports analytics. In *Proceedings of the 26th acm sigkdd international conference on knowledge discovery & data mining*, pages 3154–3162.

Tank, A., Covert, I., Foti, N., Shojaie, A., and Fox, E. B. (2021). Neural Granger causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–1.

Yıldırım, Ö., Pławiak, P., Tan, R.-S., and Acharya, U. R. (2018). Arrhythmia detection using deep convolutional neural network with long duration ECG signals. *Computers in biology and medicine*, 102:411–420.

Yuan, M. (2011). On the identifiability of additive index models. *Statistica Sinica*, pages 1901–1911.

Zhang, K., Peters, J., Janzing, D., and Schölkopf, B. (2011). Kernel-based conditional independence test and application in causal discovery. In *27th Conference on Uncertainty in Artificial Intelligence (UAI 2011)*, pages 804–813. AUAI Press.

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. (2018). DAGs with NO TEARS: Continuous optimization for structure learning. *Advances in Neural Information Processing Systems*, 31.

Zheng, X., Dan, C., Aragam, B., Ravikumar, P., and Xing, E. (2020). Learning sparse nonparametric DAGs. In *International Conference on Artificial Intelligence and Statistics*, pages 3414–3425. PMLR.

Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560.

A OTHER METHODS FOR TIME SERIES DATA

The comparison methods in our experiments come from the same model class as NTS-NOTEARS: temporal graphs with both intra-slice and inter-slice dependencies. In this section we discuss other methods for time series data, with an emphasis on neural methods (see Table 1).

Models without intra-slice edges. The neural method cMLP (Tank et al., 2021) does not have the capability to learn intra-slice edges. On datasets where the ground-truth model comprises only inter-slice edges, its performance is competitive with NTS-NOTEARS (e.g., worse F1-score on Lorenz, better F1-score on fMRI). GVAR (Marcinkevics and Vogt, 2021) also does not learn intra-slice edges, and does not explicitly model different lags. Economy-SRU (Khanna and Tan, 2019) does not estimate edges over multiple lags.

Other Methods with Intra-Slice Edges. We conducted experiments using linear methods such as VAR-LINGAM (Hyvärinen et al., 2010) and PCMCI+ with linear CI test - partial correlation test (ParCorr) (Runge et al., 2019). We exclude their results because their performance is poor for the nonlinear datasets in our experiments.

Among constraint-based methods, LPCMCI (Gerhardus and Runge, 2020) focuses on latent confounders and outputs a PAG. With nonlinear CI test, LPCMCI is computationally too expensive to be compared when the number of nodes is large. CMIknn and CMIsymb (Runge et al., 2019) are nonlinear CI tests based on conditional mutual information. Although the two CI tests are nonparametric, they are computationally expensive when the number of nodes is large, which makes them infeasible to be included in the experiments.

IDYNO (Gao et al., 2022) matches NTS-NOTEARS in terms of the properties listed in Table 1. Their work was published after we posted the arxiv version of this paper. The main difference is that IDYNO focuses on interventional data, with only one experiment on observational data with linear relationships. Other differences include the following:

1. IDYNO uses a generalized linear model. We use a nonparametric CNN to exploit the sequence topology.
2. Like DYNOTEARS, IDYNO must concatenate the data. So the original data with dimension $T \times d$ becomes $T \times d \times (K+1)$. Larger data size may cause memory problem as well as slowing down training. The CNN allows NTS-NOTEARS to use the original data without data concatenation.
3. IDYNO uses 3 MLPs for each target variable. So there are 3 times more neural nets to train compared to NTS-NOTEARS.

As no code is available for IDYNO, we used our own implementation[3]. We used the same methodology described in the paper to evaluate IDYNO with the Lorenz 96 and fMRI benchmarks. IDYNO is less accurate than NTS-NOTEARS, by about a factor of 2 in F1-score and SHD. IDYNO is also slower than NTS-NOTEARS. For example, with $d = 5$, $K = 1$ and $T \in \{200, 1200, 5000\}$ in the fMRI benchmark, IDYNO is about 7 times slower than NTS-NOTEARS. IDYNO is even slower with larger K, because it means more data to concatenate and to put in the memory.

B PROOF OF THEOREM 1

Proof. To show $F = F_0$, we will show that $F_0 \subseteq F$ and $F \subseteq F_0$.

We have:

$$F = \{f | f(X) = CNN(X; C^{(1)}, \dots, C^{(h_c)}, A^{(1)}, \dots, A^{(h_a)}), f \text{ is independent of } X_i^k\}$$

and

$$F_0 = \{f | f(X) = CNN(X; C^{(1)}, \dots, C^{(h_c)}, A^{(1)}, \dots, A^{(h_a)}), C_{i,b}^{(1),k} = 0, \forall b = \{1, \dots, m\}\}$$

[3]https://github.com/xiangyu-sun-789/IDYNO_reproduce

where $C^{(u)}$ is the kernel weights on the u-th CNN layer, $C_{i,b}^{(1),k}$ is the first-layer kernel weights in the b-th kernel connecting to input variable X_i^k, and $A^{(u)}$ is the weights on the u-th MLP layer. The bias terms are omitted as they do not affect the proof.

Also,

$$CNN(X; C^{(1)}, \dots, C^{(h_c)}, A^{(1)}, \dots, A^{(h_a)}) = \sigma(A^{(h_a)} * \sigma(\dots A^{(1)} * \sigma(C^{(h_c)} \circ \sigma(\dots \sigma(C^{(1)} \circ X)))))$$

where $*$ is matrix product, $\circ$ is the convolution operation of two matrices and σ is the activation functions.

(1) To show $F_0 \subseteq F$:

For any $f_0 \in F_0$, we have $f_0(X) = CNN(X; C^{(1)}, \dots, C^{(h_c)}, A^{(1)}, \dots, A^{(h_a)})$ where $C_{i,b}^{(1),k} = 0$ for all $b = \{1, \dots, m\}$. Therefore, $C^{(1)} \circ X$ is independent of X_i^k. Therefore, $f_0(X) = \sigma(A^{(h_a)} * \sigma(\dots A^{(1)} * \sigma(C^{(h_c)} \circ \sigma(\dots \sigma(C^{(1)} \circ X)))))$ is also independent of X_i^k. Hence, $f_0 \in F$.

(2) To show $F \subseteq F_0$:

For any $f \in F$, we have $f(X) = CNN(X; C^{(1)}, \dots, C^{(h_c)}, A^{(1)}, \dots, A^{(h_a)})$ and f is independent of X_i^k. Let $\tilde{X}$ be identical to X except $\tilde{X}_i^k = 0$. f is independent of X_i^k, similarly, f is independent of $\tilde{X}_i^k$. Therefore,

$$\begin{aligned} f(X) = f(\tilde{X}) &= CNN(\tilde{X}; C^{(1)}, \dots, C^{(h_c)}, A^{(1)}, \dots, A^{(h_a)}) \\ &= \sigma(A^{(h_a)} * \sigma(\dots A^{(1)} * \sigma(C^{(h_c)} \circ \sigma(\dots \sigma(C^{(1)} \circ \tilde{X}))))) \end{aligned} \tag{4}$$

Let $\tilde{C}^{(1)}$ be identical to $C^{(1)}$ except $\tilde{C}_{i,b}^{(1),k} = 0$ for all $b = \{1, \dots, m\}$. Let $C_{(b)}^{(1)}$ be the first-layer kernel weights of the b-th kernel. We have:

$$\begin{aligned} C_{(b)}^{(1)} \circ \tilde{X} &= \sum_{k'=1}^{K+1} \sum_{i'=1}^{d} C_{i',b}^{(1),k'} \cdot \tilde{X}_{i'}^{k'} \\ &= (\sum_{k' \neq k} \sum_{i' \neq i} C_{i',b}^{(1),k'} \cdot \tilde{X}_{i'}^{k'}) + (\sum_{i' \neq i} C_{i',b}^{(1),k} \cdot \tilde{X}_{i'}^{k}) + (\sum_{k' \neq k} C_{i,b}^{(1),k'} \cdot \tilde{X}_{i}^{k'}) + C_{i,b}^{(1),k} \cdot \tilde{X}_i^k \\ &= (\sum_{k' \neq k} \sum_{i' \neq i} C_{i',b}^{(1),k'} \cdot X_{i'}^{k'}) + (\sum_{i' \neq i} C_{i',b}^{(1),k} \cdot X_{i'}^{k}) + (\sum_{k' \neq k} C_{i,b}^{(1),k'} \cdot X_{i}^{k'}) + C_{i,b}^{(1),k} \cdot \tilde{X}_i^k \\ &= (\sum_{k' \neq k} \sum_{i' \neq i} C_{i',b}^{(1),k'} \cdot X_{i'}^{k'}) + (\sum_{i' \neq i} C_{i',b}^{(1),k} \cdot X_{i'}^{k}) + (\sum_{k' \neq k} C_{i,b}^{(1),k'} \cdot X_{i}^{k'}) + 0 \\ &= (\sum_{k' \neq k} \sum_{i' \neq i} C_{i',b}^{(1),k'} \cdot X_{i'}^{k'}) + (\sum_{i' \neq i} C_{i',b}^{(1),k} \cdot X_{i'}^{k}) + (\sum_{k' \neq k} C_{i,b}^{(1),k'} \cdot X_{i}^{k'}) + \tilde{C}_{i,b}^{(1),k} \cdot X_i^k \\ &= (\sum_{k' \neq k} \sum_{i' \neq i} \tilde{C}_{i',b}^{(1),k'} \cdot X_{i'}^{k'}) + (\sum_{i' \neq i} \tilde{C}_{i',b}^{(1),k} \cdot X_{i'}^{k}) + (\sum_{k' \neq k} \tilde{C}_{i,b}^{(1),k'} \cdot X_{i}^{k'}) + \tilde{C}_{i,b}^{(1),k} \cdot X_i^k \\ & \sum_{k'=1}^{K+1} \sum_{i'=1}^{d} \tilde{C}_{i',b}^{(1),k'} \cdot X_{i'}^{k'} = \tilde{C}_{(b)}^{(1)} \circ X \end{aligned}$$

Therefore, $C^{(1)} \circ \tilde{X} = \tilde{C}^{(1)} \circ X$. From Equation (4), we have:

$$f(X) = \sigma(A^{(h_a)} * \sigma(\dots A^{(1)} * \sigma(C^{(h_c)} \circ \sigma(\dots \sigma(\tilde{C}^{(1)} \circ X))))) = CNN(X; \tilde{C}^{(1)}, \dots, C^{(h_c)}, A^{(1)}, \dots, A^{(h_a)}) \in F_0$$

Hence, $f \in F_0$

□

C SIMULATION DETAILS

Given a graph generated by either an ER or a BA scheme, we simulate data according to one of the three identifiable SCMs:

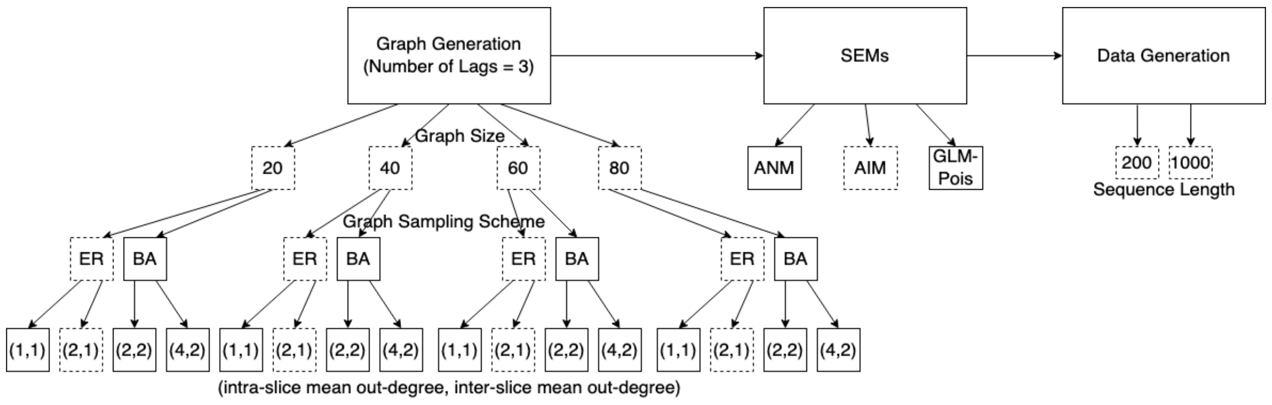


Figure 5: Visualization of the process for generating simulated training data. Besides generating the training data, the dashed boxes also indicate how the validation data was generated. A total of 48 DBNs and 96 training datasets were generated.

- Additive Noise Model (ANM) (Peters et al., 2017): $X_j^t = f_j(PA(X_j^t) \cdot \theta_1) \cdot \theta_2 + Z_j^t$, where f_j is the sigmoid function.
- Additive Index Model (AIM) (Yuan, 2011; Alquier and Biau, 2013): $X_j^t = Z_j^t + \sum_{m=1}^{3} h_m(PA(X_j^t) \cdot \theta_m)$, where $h_1 = tanh, h_2 = cos, h_3 = sin$.
- Generalized Linear Model with Poisson Distribution (GLM-Pois) (Park and Park, 2019): $X_j^t = Pois(g_j(PA(X_j^t) \cdot \theta_1) + \phi)$, where $g_j = tanh$ and ϕ is sampled uniformly from range $[1, 3]$.

$PA(X_j^t)$ denotes the parents of X_j^t. Each Z_j is a standard Gaussian noise. Each θ is sampled uniformly from range $[-2, -0.5] \cup [0.5, 2]$.

D METHOD HYPERPARAMETERS FOR SIMULATED DATA

Regarding hyperparameter searching, for compatibility purpose with constraint-based baseline PCMCI+ (Runge, 2020), we use validation sets to find hyperparameter values. We perform an extensive grid search on the hyperparameters of each method to find the sets of hyperparameters that give the best F1-scores for each method with the validation sets.

- NTS-NOTEARS
  - $\boldsymbol{\lambda_1} \in \{\mathbf{0.01}, \mathbf{0.001}\}$ for $T \in \{200, 1000\}$, respectively.
  - $\lambda_2 = 0.05$
  - $K =$ *number of lags*
  - $m = d$
  - the number of hidden layers $= 1$
  - $\boldsymbol{W}_{\textit{thres}} = \mathbf{0.3}$
- PCMCI+
  - CI test: Gaussian process regression plus distance correlation test (GPDC)
  - $\tau_{\textit{min}} = 0$
  - $\tau_{\textit{max}} =$ *number of lags*
  - $\alpha \in \{0.01, 0.05\}$ for $T \in \{500, 2000\}$, respectively
- TCDF
  - *significance* $= 0.8$
  - *learning rate* $= 0.001$
  - *epochs* $= 1000$

- $\textit{levels} = 2$
- $\textit{kernel size} = \textit{number of lags} + 1$
- $\textit{dilation coefficient} = \textit{number of lags} + 1$

- DYNOTEARS
  - $\lambda_a = \lambda_w = 0.1$
  - $p = \textit{number of lags}$
  - $\textit{weight threshold} = 0.01$

E MORE RESULTS WITH SIMULATED DATA

Besides reporting the F1-score in the main article, we also use recall, precision and SHD to evaluate the methods with simulated datasets. Please see Figure 6, 7, 8. NTS-NOTEARS achieves the best recall and SHD in the vast majority of the settings and is among the top methods in terms of precision.

F METHOD HYPERPARAMETERS FOR LORENZ 96 & FMRI BENCHMARKS

To find the hyperparameter values for the evaluation methods, we select one dataset from each benchmark as the validation sets, and perform grid search over hyperparameters to maximize the F1-socre. These hyperparameters are used for evaluation with the benchmarks. For NTS-NOTEARS, a unique set of hyperparameters is used for both benchmarks:

- NTS-NOTEARS
  - $\lambda_1^1 = 0.001$, $\lambda_1^2 = 0.1$
  - $\lambda_2 = 0.01$
  - $K = \textit{number of lags}$
  - $m = 2d$
  - the number of hidden layers $= 1$
  - $\boldsymbol{W}_{\textit{thres}} = \mathbf{0.5}$

For the Lorenz 96 benchmark:

- PCMCI+
  - CI test: GPDC
  - $\tau_{\textit{min}} = 0$
  - $\tau_{\textit{max}} = \textit{number of lags}$
  - $\alpha = 0.005$
- TCDF
  - $\textit{significance} = 1.5$
  - $\textit{learning rate} = 0.001$
  - $\textit{epochs} = 1000$
  - $\textit{levels} = 3$
  - $\textit{kernel size} = \textit{number of lags} + 1$
  - $\textit{dilation coefficient} = \textit{number of lags} + 1$
- DYNOTEARS
  - $\lambda_w = 0.1$
  - $\lambda_a = 0.01$
  - $p = \textit{number of lags}$
  - $\textit{weight threshold} = 0.1$

For the fMRI benchmark:

- PCMCI+
  - CI test: GPDC
  - $\tau_{min} = 0$
  - $\tau_{max} = \textit{number of lags}$
  - $\alpha = 0.001$
- TCDF
  - $\textit{significance} = 0.5$
  - $\textit{learning rate} = 0.01$
  - $\textit{epochs} = 2000$
  - $\textit{levels} = 2$
  - $\textit{kernel size} = \textit{number of lags} + 1$
  - $\textit{dilation coefficient} = \textit{number of lags} + 1$
- DYNOTEARS
  - $\lambda_w = 0.1$
  - $\lambda_a = 0.1$
  - $p = \textit{number of lags}$
  - $\textit{weight threshold} = 0.1$

G MORE RESULTS WITH LORENZ 96 & FMRI BENCHMARKS

Besides reporting the F1-score in the main article, we also use SHD, precision and recall to evaluate the methods with benchmark datasets. Please see Table 4, 5, 6. NTS-NOTEARS achieves the best SHD, recall and precision with both benchmarks. Around 20 combinations of hyperparameter values were used for each method to compute the AUROC in Table 3.

Table 4: **Mean SHDs (± SE)** computed with the benchmark datasets.

Method	Lorenz 96	fMRI
DYNOTEARS	20.4 (± 1.992)	10.8 (± 0.645)
TCDF	58.0 (± 1.789)	9.0 (± 0.600)
PCMCI+	44.6 (± 2.492)	8.1 (± 0.624)
NTS-NOTEARS	**0.6 (± 0.358)**	**6.6 (± 0.651)**

Table 5: **Mean recalls (± SE)** computed with the benchmark datasets.

Method	Lorenz 96	fMRI
DYNOTEARS	0.760 (± 0.024)	0.460 (± 0.016)
TCDF	0.308 (± 0.013)	0.246 (± 0.043)
PCMCI+	0.495 (± 0.032)	0.406 (± 0.043)
NTS-NOTEARS	**0.993 (± 0.004)**	**0.516 (± 0.014)**

Table 6: **Mean precisions ($\pm$ SE)** computed with the benchmark datasets.

Method	Lorenz 96	fMRI
DYNOTEARS	0.981 ($\pm$ 0.007)	0.508 ($\pm$ 0.043)
TCDF	0.906 ($\pm$ 0.035)	0.610 ($\pm$ 0.098)
PCMCI+	0.903 ($\pm$ 0.006)	0.754 ($\pm$ 0.070)
NTS-NOTEARS	**1.000 ($\pm$ 0.000)**	**0.839 ($\pm$ 0.062)**

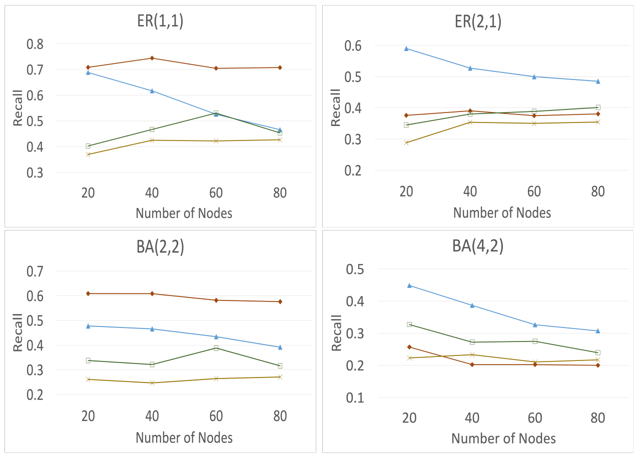
H REAL-WORLD ICE HOCKEY DATASET

Please see Table 7 for data description and Figure 10 for data distribution plots. A nominal variable with K values can be converted to K binary variables using one-hot encoding. The following hyperparameter values are used in Section 6.3. With these hyperparameter values, NTS-NOTEARS and DYNOTEARS generate graphs with the same number of edges.

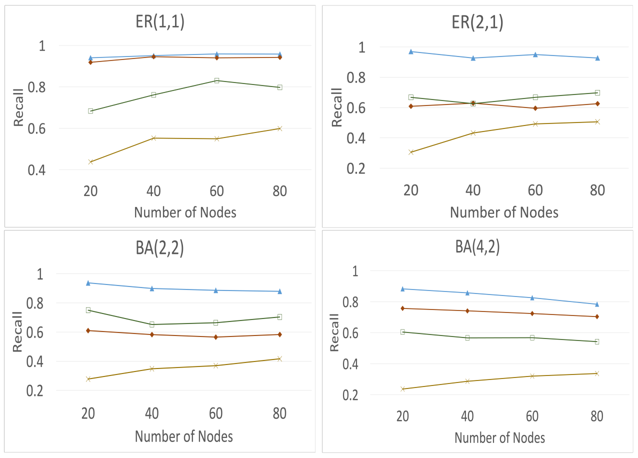
- NTS-NOTEARS
  - $\boldsymbol{\lambda_1 = 0.001}$
  - $\lambda_2 = 0.005$
  - $K = 1$
  - $m = d$
  - the number of hidden layers $= 1$
  - $\boldsymbol{W_{\textit{thres}} = 0.5}$
- DYNOTEARS
  - $\lambda_w = 0.01$
  - $\lambda_a = 0.01$
  - $p = 1$
  - *weight threshold* $= 0.03$

Table 7: The variables in the ice hockey dataset.

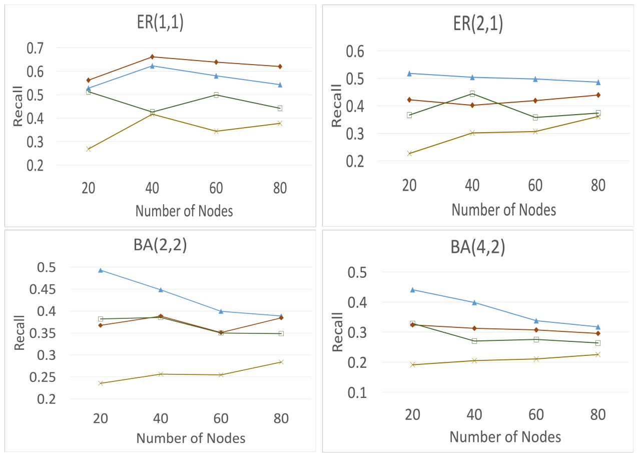
Variables	Type	Range
time remaining in seconds	continuous	[0, 3600]
adjusted x coordinate of puck	continuous	[-100, 100]
adjusted y coordinate of puck	continuous	[-42.5, 42.5]
score differential	categorical	$(-\infty, +\infty)$
manpower situation	categorical	{short handed, even strength, power play}
x velocity of puck	continuous	$(-\infty, +\infty)$
y velocity of puck	continuous	$(-\infty, +\infty)$
event duration	continuous	$[0, +\infty)$
angle between puck and net	continuous	$[-\pi, +\pi]$
home team taking possession	binary	{true, false}
shot	binary	{true, false}
goal	binary	{true, false}



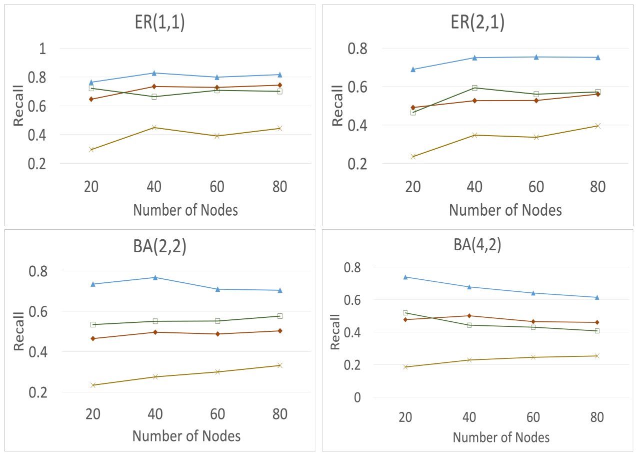
(a) Additive Index Model, $T = 200$



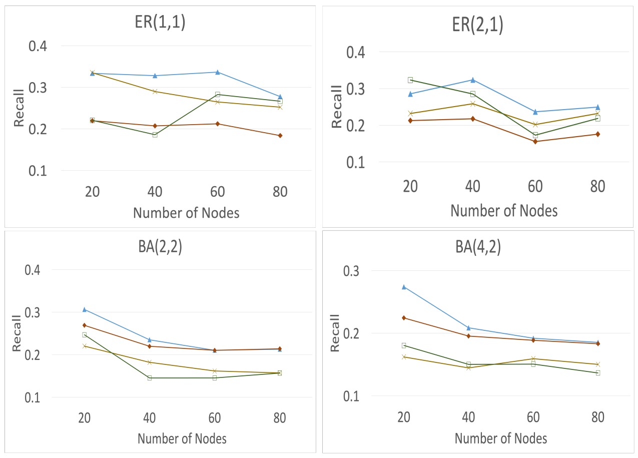
(b) Additive Index Model, $T = 1000$



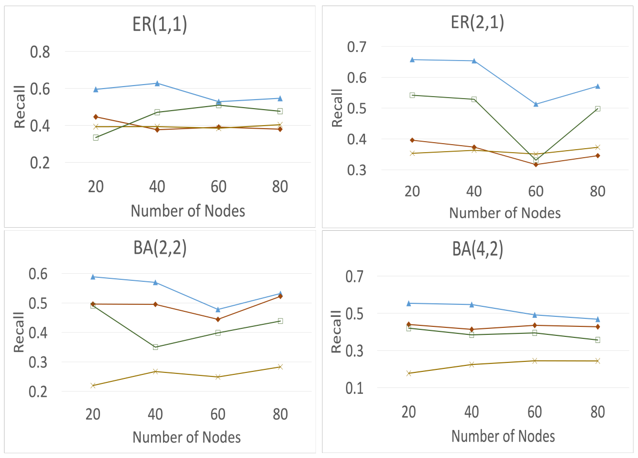
(c) Additive Noise Model, $T = 200$



(d) Additive Noise Model, $T = 1000$



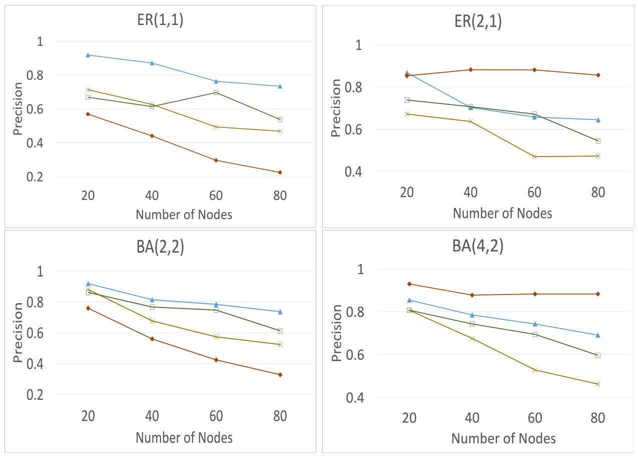
(e) Generalized Linear Model with Poisson Distribution, $T = 200$



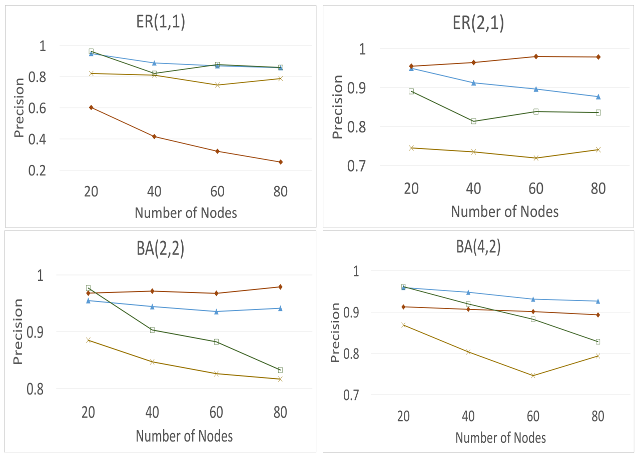
(f) Generalized Linear Model with Poisson Distribution, $T = 1000$

▲NTS-NOTEARS ◆DYNOTEARS ×TCDF ⊟PCMCI+

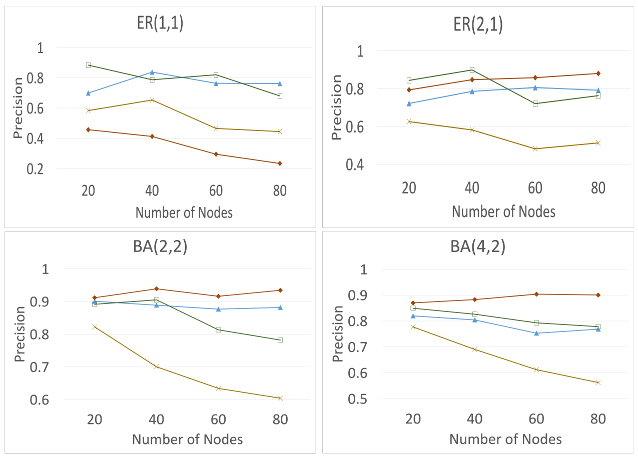
Figure 6: **Mean recalls** over 10 datasets for each setting with simulated data. Higher Recall is better. The number of lags $= 3$.



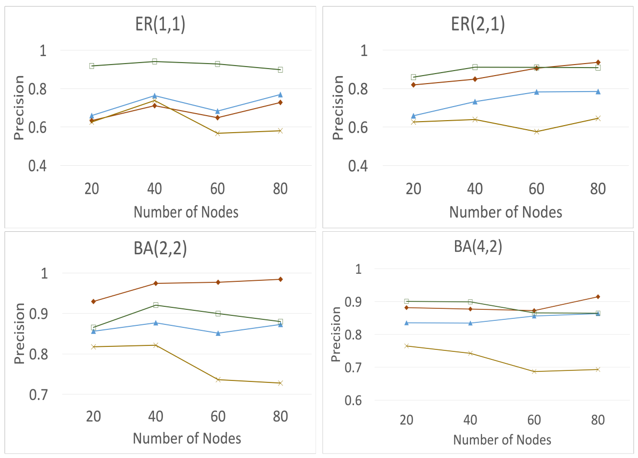
(a) Additive Index Model, $T = 200$



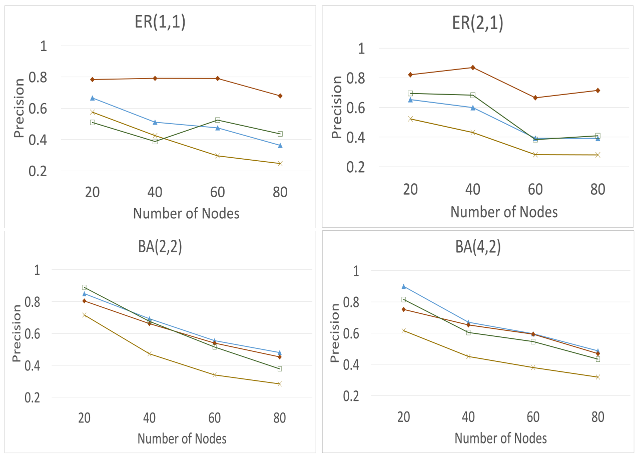
(b) Additive Index Model, $T = 1000$



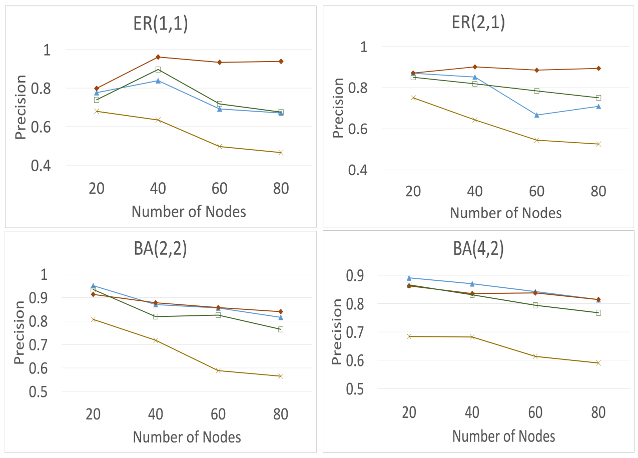
(c) Additive Noise Model, $T = 200$



(d) Additive Noise Model, $T = 1000$



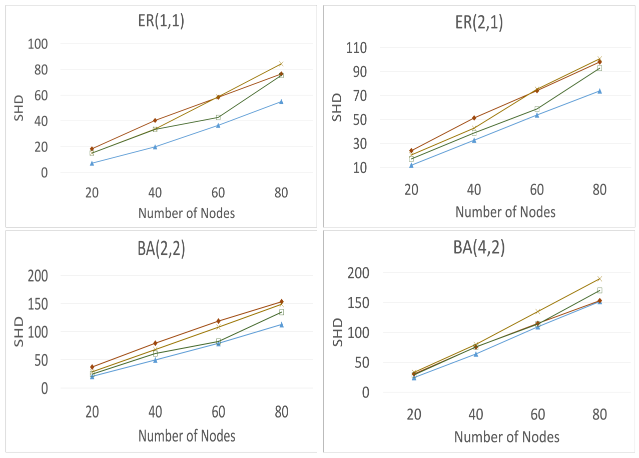
(e) Generalized Linear Model with Poisson Distribution, $T = 200$



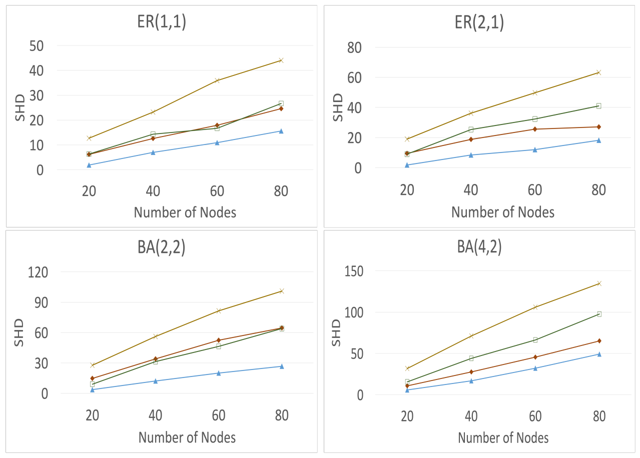
(f) Generalized Linear Model with Poisson Distribution, $T = 1000$

NTS-NOTEARS
DYNOTEARS
TCDF
PCMCI+

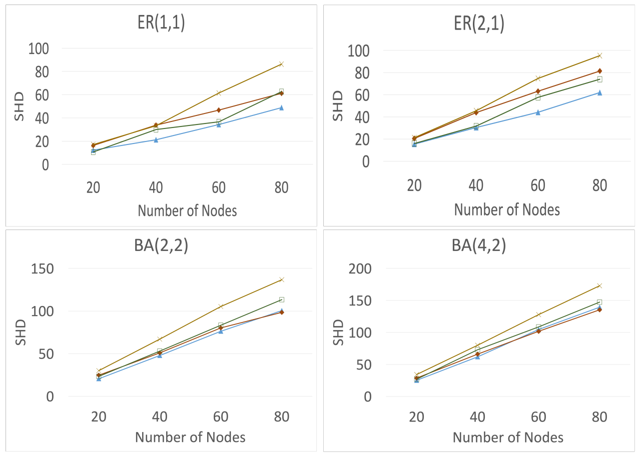
Figure 7: **Mean precisions** over 10 datasets for each setting with simulated data. Higher Precision is better. The number of lags = 3.



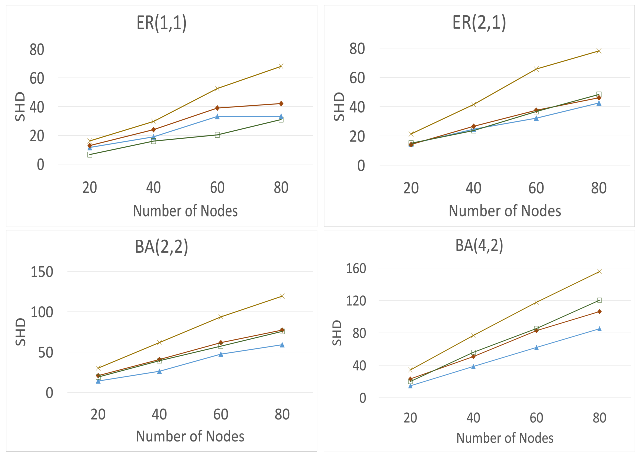
(a) Additive Index Model, $T = 200$



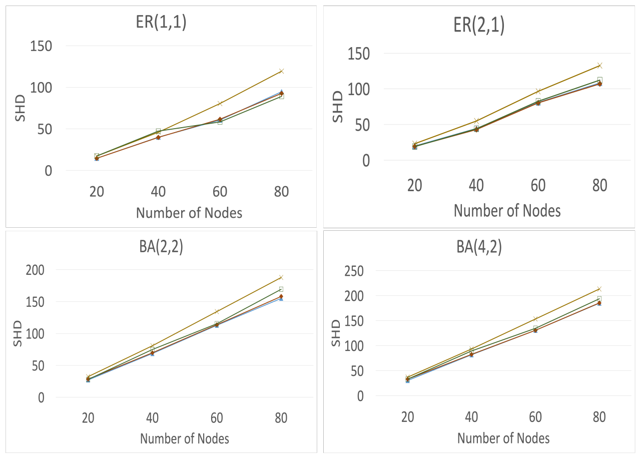
(b) Additive Index Model, $T = 1000$



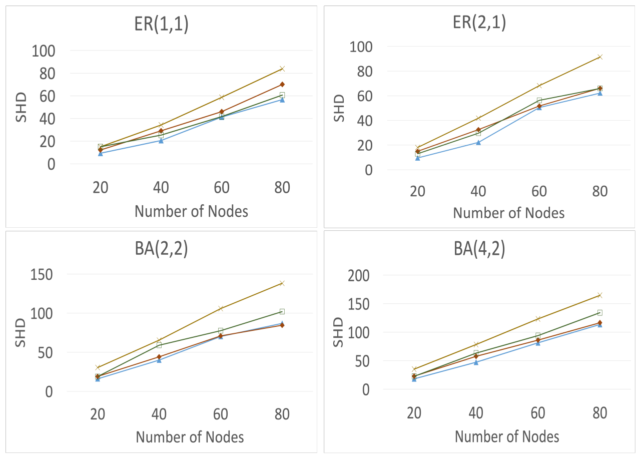
(c) Additive Noise Model, $T = 200$



(d) Additive Noise Model, $T = 1000$



(e) Generalized Linear Model with Poisson Distribution, $T = 200$



(f) Generalized Linear Model with Poisson Distribution, $T = 1000$

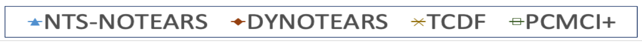
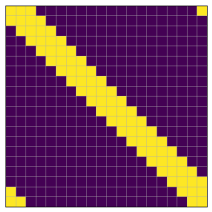


Figure 8: **Mean SHDs** over 10 datasets for each setting with simulated data. Lower SHD is better. The number of lags $= 3$.


(a) The ground-truth DBN from the Lorenz 96 benchmark

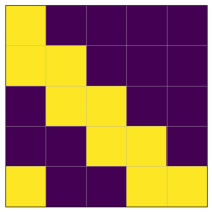

(b) The ground-truth DBN from the fMRI benchmark

Figure 9: DBNs showing edges from the lag pointing to the instantaneous time step.

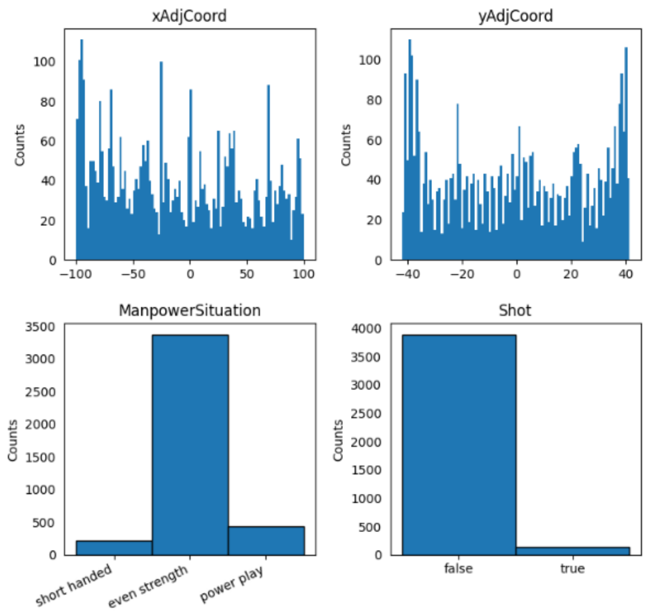
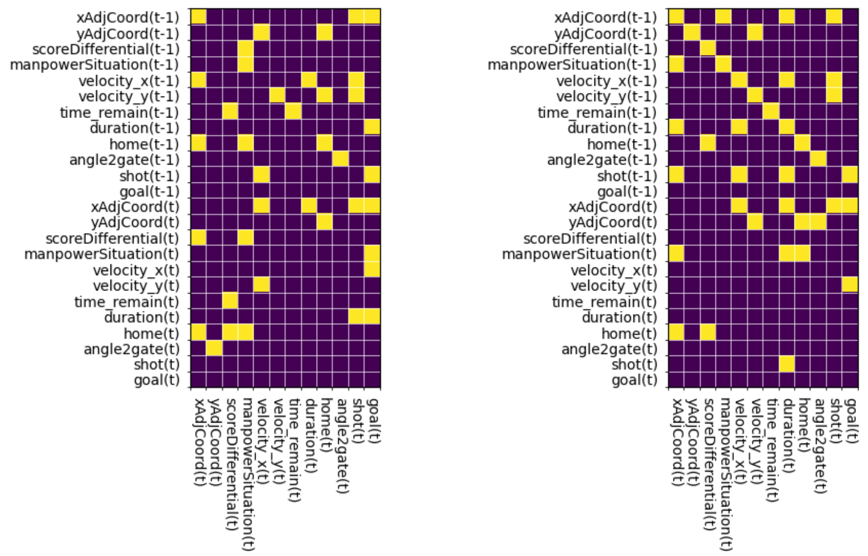


Figure 10: The distributions of two non-Gaussian continuous variables and two discrete variables in the ice hockey dataset.



(a) NTS-NOTEARS

(b) DYNOTEARS

Figure 11: The DBNs estimated by NTS-NOTEARS and DYNOTEARS with real-world ice hockey data.