Learning Identifiable Structures Helps Avoid Bias in DNN-based Supervised Causal Learning

Jiaru Zhang[*]
Shanghai Jiao Tong University
jiaruzhang@sjtu.edu.cn

Rui Ding[†]
Microsoft
juding@microsoft.com

Qiang Fu
Microsoft
qifu@microsoft.com

Huang Bojun
Sony Research
bojhuang@gmail.com

Zizhen Deng
Peking University
dengzizhen557@outlook.com

Yang Hua
Queen's University Belfast
y.hua@qub.ac.uk

Haibing Guan
Shanghai Jiao Tong University
hbguan@sjtu.edu.cn

Shi Han
Microsoft
shihan@microsoft.com

Dongmei Zhang
Microsoft
dongmeiz@microsoft.com

Abstract

Causal discovery is a structured prediction task that aims to predict causal relations among variables based on their data samples. Supervised Causal Learning (SCL) is an emerging paradigm in this field. Existing Deep Neural Network (DNN)-based methods commonly adopt the "Node-Edge approach", in which the model first computes an embedding vector for each variable-node, then uses these variable-wise representations to concurrently and independently predict for each directed causal-edge. In this paper, we first show that this architecture has some systematic bias that cannot be mitigated regardless of model size and data size. We then propose SiCL, a DNN-based SCL method that predicts a skeleton matrix together with a v-tensor (a third-order tensor representing the v-structures). According to the Markov Equivalence Class (MEC) theory, both the skeleton and the v-structures are *identifiable* causal structures under the canonical MEC setting, so predictions about skeleton and v-structures do not suffer from the identifiability limit in causal discovery, thus SiCL can avoid the systematic bias in Node-Edge architecture, and enable consistent estimators for causal discovery. Moreover, SiCL is also equipped with a specially designed pairwise encoder module with a unidirectional attention layer to model both internal and external relationships of pairs of nodes. Experimental results on both synthetic and real-world benchmarks show that SiCL significantly outperforms other DNN-based SCL approaches.

[*]The work was done during his internship at Microsoft Research Asia.

[†]Corresponding author.

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

1 INTRODUCTION

Causal discovery seeks to infer causal structures from an observational data sample. Supervised Causal Learning (SCL) (Dai et al., 2023; Ke et al., 2023; Ma et al., 2022) is an emerging paradigm in this field. The basic idea is to consider causal discovery as a *structured prediction* task, and to train a prediction model using supervised learning techniques. At training time, a training dataset comprising a variety of causal mechanisms and their associated data samples is generated. The prediction model is then trained to take such a data sample as input, and to output predictions about the causal mechanism behind the data sample. Compared to traditional rule-based or unsupervised methods (Glymour et al., 2019), the SCL method has demonstrated strong empirical performance (Dai et al., 2023; Ma et al., 2022), as well as robustness against sample size and distribution shift (Ke et al., 2023; Lorch et al., 2022).

Deep Neural Network (DNN)-based SCL employs DNN as the prediction model. It allows end-to-end training, removing the need for manual feature engineering. Additionally, it can handle both continuous and discrete data types effectively, and can learn latent representations. A specific DNN architecture, first introduced by Lorch et al. (2022), is particularly popular in recent DNN-based SCL works. The model first transforms the given data sample into a set of node-wise feature vectors, each representing an individual variable (corresponding to a node in the associated causal graph). Based on these node-wise features, the model then outputs a weighted adjacency matrix A, where $A_{ij} \in [0, 1]$ is an estimated probability for the directed edge $i \to j$ (meaning that i is a direct cause of j). Finally, the adjacency matrix of an inferred causal graph G is obtained as a Bernoulli sample of A, where each entry $G_{ij} \in \{0, 1\}$ is sampled *independently*, following probability A_{ij}. For convenience, we call such a model architecture as the "Node-Edge" architecture, as the representation is learned for individual nodes and the probability is estimated and sampled for individual directed edges.

Despite its popularity and encouraging results (Lorch et al., 2022; Zhu et al., 2020; Charpentier et al., 2022; Varambally et al., 2024), we identify two limitations for the Node-Edge approach:

First, the Node-Edge architecture imposes a fundamental bias in the inferred causal relations. Specifically, given an observational data sample D, the existence of a directed causal edge $i \to j$ may *necessarily* depend on the existence of other edges. But the existing Node-Edge models predict each edge separately and independently, so the probability prediction A_{ij} made by such models is only conditioned on the input sample D, not on the sampling result of other entries of A, thereby failing to capture the crucial inter-edge dependency in its probability estimation.

As a simple example, a Node-Edge model maintaining the possibility of both $G_1 : X \to T \to Y$ and $G_2 : X \leftarrow T \leftarrow Y$ would necessarily have a non-zero probability to output the edges $X \to T$ and $T \leftarrow Y$, thus cannot rule out the possibility of $G_3 : X \to T \leftarrow Y$, even though G_3 is impossible to be the groundtruth causal graph behind a data sample D compatible with G_1 and G_2 (Verma and Pearl, 1990). Crucially, there is no way to tell G_1 from G_2 based on observational data in general cases (Andersson et al., 1997; Meek, 1995b). It means that for any Node-Edge model to be sound, it has to maintain the possibility of both G_1 and G_2 (when observing a data sample compatible with any of them), leading to an inevitable error probability to output the impossible graph G_3 on the other hand.

Second, the Node-Edge architecture does not explicitly represent the features about node pairs, which we argue are essential for observational causal discovery. For example, a causal edge $X \to Y$ can exist only if the node pair $\langle X, Y \rangle$ demonstrates *persistent dependency* (Ma et al., 2022; Spirtes et al., 2000), meaning that X and Y remain statistically dependent regardless of conditioning on any subset of other variables. As another example, for causal DAGs, a sufficient condition to determine the causal direction between a persistently dependent node pair $\langle X, Y \rangle$ is that X and Y exhibits *orientation asymmetry*, meaning that there exists a third variable Z such that X is persistently dependent to Z but Y can become independent to Z conditioned on a variable-set $\mathbf{S} \not\ni X$ (or vice versa). A feature like persistent dependency or orientation asymmetry is, in its nature, a collective property of a node pair, but not of any individual node alone.

To address these limitations, in this paper, we propose a novel DNN-based SCL approach, called Supervised Identifiable Causal Learning (SiCL). The neural network in SiCL does not seek to predict the probabilities of directed edges, but tries to predict a skeleton matrix together with v-tensor, a third-order tensor representing the v-structures. According to the Markov Equivalence Class (MEC) theory of causal discovery, skeleton and v-structures are *identifiable* causal structures under the canonical MEC setting (while the directed edges are not), so predictions about skeleton and v-structures do not suffer from the (non-)identifiability limit. By leveraging this insight, our theory-inspired DNN architecture completely avoids the systematic bias in edge-prediction models as previously discussed, and enables *consistent* neural-estimators[1] for causal discovery. Moreover, SiCL is also equipped with a specially designed pairwise encoder module with a unidirectional attention layer. With both node features and node-pair features as the layer input, it can model both internal and external relationships of pairs of nodes. Experimental results on both synthetic and real-word benchmarks show that SiCL can effectively address the two abovementioned limitations, and the resulted SiCL solution significantly outperforms other DNN-based SCL approaches with more than 50% performance improvement in terms of SHD (Structural Hamming Distance) on the real-world Sachs data. The codes are publicly available at `https://github.com/microsoft/reliableAI/tree/main/causal-kit/SiCL`.

[1]Recall that a statistical estimator is *consistent* if it converges to the groundtruth given infinite data.

2 BACKGROUND AND RELATED WORK

A Causal Graphical Model is defined by a joint probability distribution P over multiple random variables and a DAG G. Each node X_i in G represents a variable in P, and a directed edge $X_i \to X_j$ represents a direct cause-effect relation from X_i to X_j. A causal discovery task generally asks to infer about G from an i.i.d. sample of P.

However, there is a well-known identifiability limit for causal discovery. In general, the causal DAG is only identifiable up to an equivalence class. Studies of this identifiability limit under a canonical assumption setting have led to the well-established MEC theory (Frydenberg, 1990; Verma and Pearl, 1990). We call a causal feature, *MEC-identifiable*, if the value of this feature is invariant among the equivalence class under the canonical MEC assumption setting. It is known that such MEC-identifiable features include the skeleton and the set of v-structures, which we briefly present in the following.

A *skeleton* E defined over the data distribution P is an undirected graph where an edge exists between X_i and X_j if and only if X_i and X_j are always dependent in P, i.e., $\forall Z \subseteq \{X_1, X_2, \cdots, X_d\} \setminus \{X_i, X_j\}$, we have $X_i \not\perp X_j | Z$. Under mild assumptions (such as that P is Markovian and faithful to the DAG G; see details in Appendix Sec. A1.1), the skeleton is the same as the corresponding undirected graph of the DAG G (Spirtes et al., 2000). A triple of variables $\langle X, T, Y \rangle$ is an *Unshielded Triple (UT)* if X and Y are both adjacent to T but not adjacent to each other in (the skeleton of) G. It becomes a *v-structure* denoted as $X \to T \leftarrow Y$ if the directions of the edges are from X and Y to T.

Two graphs are Markov equivalent if and only if they have the same skeleton and v-structures. The *Markov equivalence class (MEC)* can be represented by a *Completed Partially Directed Acyclic Graph (CPDAG)* consisting of both directed and undirected edges. We use $CPDAG(G)$ to denote the CPDAG derived from G. According to the theorem of Markov completeness (Meek, 1995b), we can only identify a causal graph up to its MEC, i.e., the CPDAG, unless additional assumptions are made (see the remark below). This means that each (un)directed edge in $CPDAG(G)$ indicates a (non)identifiable causal relation.

Remark: The MEC-based identifiability theory is applicable in the general-case setting, when we take into account all possible distributions P. It is known this identifiability limit could be broken (i.e., an undirected edge in the CPDAG could be oriented) *if* we assume that the data follows some special class of distributions, e.g., linear non-Gaussians, additive noise models, post-nonlinear or location-scale models (Peters et al., 2014; Shimizu et al., 2011; Zhang and Hyvärinen, 2009; Immer et al., 2023). These assumptions are sometimes hard to verify in practice, so this paper considers the general-case setting. More discussions on the identifiability and causal assumptions are provided in Appendix Sec. A2.

2.1 Related Work

In traditional methods of causal discovery, constraint-based methods are mostly related to our work. They aim to identify the DAG that is consistent with inter-variable conditional independence constraints. These methods first identify the skeleton and then conduct orientation based on v-structure identification (Yu et al., 2016). The output is a CPDAG which represents the MEC. Notable algorithms in this category include PC (Spirtes et al., 2000), along with variations such as Conservative-PC (Ramsey et al., 2012), PC-stable (Colombo et al., 2014), and Parallel-PC (Le et al., 2016). Compared to constraint-based methods, both our approach and theirs are founded upon the principles of MEC theory for estimating skeleton and v-structures. However, whereas traditional methods rely on symbolic reasoning based on explicit constraints, we employ DNNs to capture the essential causal information intricately linked with these constraints.

Score-based methods aim to find an optimal DAG according to a predefined score function, subject to combinatorial constraints. These methods employ specific optimization procedures such as forward-backward search GES (Chickering, 2002), hill-climbing (Koller and Friedman, 2009), and integer programming (Cussens, 2011). Continuous optimization methods transform the discrete search procedure into a continuous equality constraint. NOTEARS (Zheng et al., 2018) formulates the acyclic constraint as a continuous equality constraint and is further extended by DAG-GNN (Yu et al., 2019), DECI (Geffner et al., 2022) to support non-linear causal relations. DECI (Geffner et al., 2022) is a flow-based model which can perform both causal discovery and inference on non-linear additive noise data. Recently, ENCO (Lippe et al., 2022) is proposed as a continuous optimization method where the edge orientation is modeled as a separate parameter to maintain the acyclicity. It is guaranteed to converge to the correct graph if interventions on all variables are available. RL-BIC (Zhu et al., 2020) utilizes Reinforcement Learning to search for the optimal DAG. These methods can be viewed as unsupervised since they do not access additional datasets associated with ground truth causal relations. We refer to Glymour et al. (2019); Vowels et al. (2022) for a thorough exploration of this literature.

SCL begins from orienting edges in the bivariate cases under the functional causal model formalism. Methods such as RCC (Lopez-Paz et al., 2015) and NCC (Lopez-Paz et al., 2017) have outperformed unsupervised approaches like ANM (Hoyer et al., 2008) or IGCI (Janzing et al., 2012). For multivariate cases, ML4S (Ma et al., 2022) proposes a supervised approach specifically for skeleton learning. Complementary to ML4S, ML4C (Dai et al., 2023) takes both data and skeleton as input and classifies unshielded triples as either v-structures or non-v-structures. Petersen et al. (2023) proposes a SLdisco method, utilizing SCL approach to address some limitations of PC and GES.

DNN-based SCL has emerged as a prominent approach for enabling end-to-end causal learning. Two notable works in this line, namely AVICI (Lorch et al., 2022) and CSIvA (Ke et al., 2023), introduced an alternating attention mechanism to enable permutation invariance across samples and variables. Both methods learn individual representation for each node, which is then used to predict directed edges. Among them, AVICI considers the task of predicting DAG from observational data and adopts exactly the Node-Edge architecture, hence suffers from the issues as discussed in Sec. 1. On the other hand, CSIvA requires additional interventional data as input to identify the full DAG, and applies an autoregressive DNN architecture where edges are predicted sequentially by multiple inference runs. Therefore, this autoregressive approach incurs very high inference cost due to the quadratic number of model runs required (w.r.t. the number of variables in question), as we experimentally verify in Appendix Sec. A8.4. In contrast, the method proposed in this paper only requires a single run of the DNN model. Besides that, our method also differs from both AVICI and CSIvA in terms of the usage of pairwise embedding vectors.

3 LIMITATIONS OF THE NODE-EDGE ARCHITECTURE

The Node-Edge architecture is common and has been adopted to generate the output DAG G in the literature (Lorch et al., 2022; Zhu et al., 2020; Charpentier et al., 2022; Varambally et al., 2024). In this architecture, each entry G_{ij} in the DAG is independently sampled from A_{ij}, an entry in the adjacency matrix A. This entry A_{ij} represents the probability that i directly causes j. We introduce a simple yet effective example setting with only three variables X, Y, and T to reveal its limitation.

Considering a simulator that generates DAGs with equal probability from two causal models: In model 1, the causal graph is $G_1 : X \to T \to Y$, and the variables follow $X \sim \mathcal{N}(0,1)$, $T = X + \mathcal{N}(0,1)$, $Y = T + \mathcal{N}(0,1)$. In model 2, the causal graph is $G_2 : X \leftarrow T \leftarrow Y$, and the variables follow $Y = \mathcal{N}(0,3)$, $T = \frac{2}{3}Y + \mathcal{N}(0, \frac{2}{3})$, $X = 0.5T + \mathcal{N}(0, 0.5)$. In this case, data samples coming from both causal models follow the same joint distribution, which makes G_1 and G_2 inherently indistinguishable (from observational data sample).

More importantly, when the fully-directed causal DAGs are used as the learning target (as the Node-Edge approach does), an optimally trained neural network will predict 0.5 probabilities on the directions of the two edges $X - T$ and $T - Y$. As a result, with 0.25 probability the graph sampling outcome would be $X \to T \leftarrow Y$ (see Fig. A4 in the Appendix). This error probability is rooted from the fact that the Bernoulli sampling of the edge $X \to T$ is not conditioned on the sampling result of the edge $T \leftarrow Y$. Consequently, it is a bias that cannot be avoided even if the DNN has perfectly modeled the *marginal probability* of each edge (marginalized over other edges) given input data.

We further find that 0.25 is not the worst-case error rate yet. Formally, for a distribution Q over a set of graphs, we define the graph distribution where the edges are independently sampled from the marginal distribution as $M(Q)$, i.e., for any causal edges e_1 and e_2, $P_{G \sim Q}(e_1 \in G) = P_{G \sim M(Q)}(e_1 \in G) = P_{G \sim M(Q)}(e_1 \in G | e_2 \in G)$. In general, a Node-Edge model optimally trained on data samples D coming from the distribution Q will essentially learn to predict $M(Q)$ (when given the same data samples D at test time). The following proposition shows that for causal graphs with star-shaped skeleton, with a chance of 26.42% the graph sampled from the marginal distribution $M(Q)$ would be incorrect.

Proposition 3.1. *Let $\mathcal{G}_n$ be the set of graphs with $n+1$ nodes where there is a central node y such that (1) every other node is connected to y, (2) there is no edge between the other nodes, and (3) there is at most one edge pointing to y. We have*

$$\sup_n \max_Q P_{G \sim M(Q)}(G \notin \mathcal{G}_n) = 1 - \frac{2}{e} \approx 0.2642. \quad (1)$$

The proof is provided in Appendix Sec. A6. It indicates that an edge-predicting neural network could suffer from an inevitable error rate of 0.2642 even if it is perfectly trained. In contrast, models that predict skeleton and v-structures would have a theoretical asymptotic guarantee of the consistency under canonical assumption. The details, proof and relevant discussions are provided in Appendix Sec. A1.

4 THE SICL METHOD

In light of the limitations as discussed, we propose a new DNN-based SCL method in this section, named **SiCL** (**S**upervised **i**dentifiable **C**ausal **L**earning).

4.1 Overall Workflow

Following the standard DNN-based SCL paradigm, the core inference process is implemented as a DNN. The DNN takes a data sample encoded by a matrix $D \in \mathbb{R}^{n \times d}$ as input, with d being the number of observable variables and n being the number of observations. In contrast to previous Node-Edge approaches, the SiCL method does not use the DNN to directly predict the causal graph, but instead seeks to predict the skeleton and the v-structures of the causal graph, which amount to the *MEC-identifiable* causal structures as mentioned previously.

Specifically, our DNN outputs two objects: (1) a skeleton prediction matrix $S \in [0,1]^{d \times d}$ where S_{ij} models the conditional probability of the existence of the *undirected* edge $X_i - X_j$, conditioned on the input data sample D, and (2) a v-structure prediction tensor $V \in [0,1]^{d \times d \times d}$ where V_{ijk} models the conditional probability of the existence of the v-structure component $X_j \rightarrow X_i \leftarrow X_k$, again conditioned on D. In our implementation, S and V are generated by two separate sub-networks, called Skeleton Predictor Network (SPN) and V-structure Predictor Network (VPN), respectively. To further address the limitation of only having node-wise features for Node-Edge models, we propose to equip SPN and VPN with Pairwise Encoder modules to explicitly capture node-pair-wise features.

Based on the skeleton prediction matrix S and v-structure prediction tensor V, we infer the skeleton and v-structures of the causal graph; from the two we can determine a unique CPDAG. The CPDAG encodes a Markov equivalence class, from which we can pick up a graph instance as the prediction of the causal DAG (if needed). Figure 1 provides a diagram of the overall inference workflow of SiCL, and a pseudo-code of the workflow is given by Algorithm A1 in Appendix.

Parameters of the SPN and VPN are trained following the standard supervised learning procedure. In our implementation, we only use synthetic training data, which is relatively easy to obtain, and yet could often lead to strong performance on real-world workloads (Ke et al., 2023).

In the following, we elaborate the DNN architecture, the learning targets, as well as the post-processing procedure used in the SiCL method.

4.2 Feature Extraction

Input Processing and Node Feature Encoder. Given input data sample, which is a matrix $D \in \mathbb{R}^{n \times d}$, the input processing module contains a linear layer for continuous input data or an embedding layer for discrete input data, yielding the raw node features $\mathcal{F}_{il}^{raw}$ for each node i in each observation l. After that, we employ a node feature encoder to further process the raw node features into the final node features $\mathcal{F}_{il}$. Similar to previous papers (Ke et al., 2023; Lorch et al., 2022), the node feature encoder is a transformer-like network comprising attention layers over the observation dimension and the node dimension alternately, which naturally maintains permutation equivalence across both variable and data dimension because of the intrinsic symmetry of attention operations. More details about the node feature encoder are presented in Appendix Sec. A4 due to page limit.

Pairwise Encoder. Given node features $\mathcal{F} \in \mathbb{R}^{d \times n \times h}$ for all the d nodes, the goal of pairwise encoder is to encode their pairwise relationships by d^2 pairwise features, represented as a tensor $\mathcal{P} \in \mathbb{R}^{d \times d \times n \times h}$, where $\mathcal{P}_{ijl} \in \mathbb{R}^h$ is a pairwise feature corresponding to the node pair (i,j) in observation l. As argued in Sec. 1, both "internal" information (i.e., the pairwise relationship) and "external information (e.g., the context of the conditional separation set) of node pairs are needed to capture persistent dependency and orientation asymmetry. Our pairwise encoder module is designed to model the internal relationship via node feature concatenation and the non-linear mapping by MLP. On the other hand, we employ attention operations within the pairwise encoder to capture the contextual relationships (including persistent dependency and orientation asymmetry).

More specifically, the pairwise encoder module consists of the following parts (see Appendix Fig. A3 for diagrammatic illustration):

1. *Pairwise Feature Initialization.* The initial step is to concatenate the node features from the previous node feature encoder module for every pair of nodes. Subsequently, we employ a three-layer MLP to convert each concatenated vector $\mathcal{P}_{ijl} \in \mathbb{R}^{2h}$ to an h-dimensional raw pairwise feature, i.e., $\mathcal{P}_{ijl}^1 = \text{MLP}([\mathcal{F}_{il}; \mathcal{F}_{jl}])$. It is designed to capture the intricate relations that exist inside the pairs of nodes.

2. *Unidirectional Multi-Head Attention.* In order to model the external information, we employ an attention mechanism where the query is composed of the aforementioned d^2 h-dimensional raw pairwise features, while the keys and values consist of h-dimensional features of d individual nodes, i.e.,

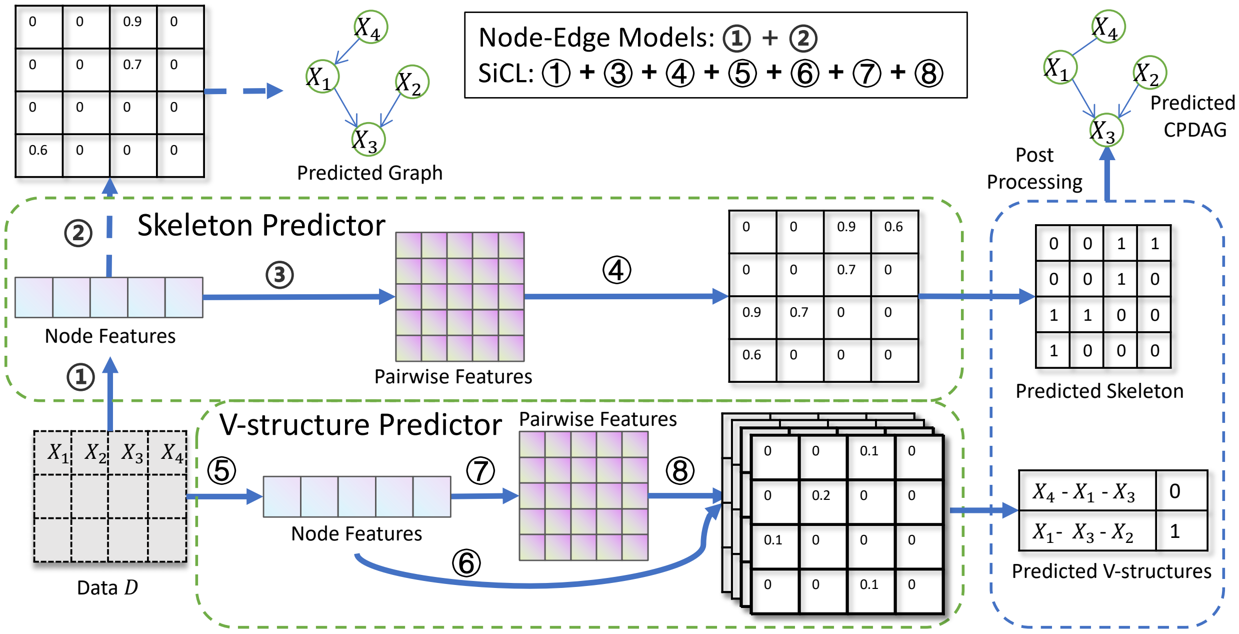


Figure 1: The inference workflow of SiCL.

$\mathcal{P}^2$ = MultiHeadAttention($\mathcal{P}^1, \mathcal{F}, \mathcal{F}$). Note that, this attention operation is unidirectional, which means we only calculate cross attention from raw pairwise features $\mathcal{P}^1$ to node features F. This design is meant to capture both pair-wise and node-wise information (as both are critical to model the causality, as discussed in Sec. 1) while at the same time to maintain a reasonable computational cost.

3. *Final Processing.* Following the widely-adopted transformer architecture, we incorporate a residual structure and a dropout layer after the previous part, i.e., $\mathcal{P}^3 = \text{Norm}(\mathcal{P}^1 + \mathcal{P}^2)$. Finally, we introduce a three-layer MLP to further capture intricate patterns and non-linear relationships between the input embeddings, as well as to more effectively process the information from the attention mechanism: $\mathcal{P} = \text{Norm}(\text{MLP}(\mathcal{P}^3) + \mathcal{P}^3)$. It yields the final pairwise feature tensor $\mathcal{P} \in \mathbb{R}^{d \times d \times n \times h}$.

4.3 Learning Targets

As mentioned above, our learning target is a combination of the skeleton and the set of v-structures, which together represent an MEC. Two separate neural (sub-)networks are trained for these two targets.

Skeleton Prediction. As the persistent dependency between pairs of nodes determines the existence of edges in the skeleton, the pairwise features correspond to edges in the skeleton naturally. Therefore, for the skeleton learning task, we initially employ a max-pooling layer over the observation dimension to obtain a single vector $\mathcal{S}_{ij} \in \mathbb{R}^h$ for each pair of nodes, i.e., $\mathcal{S}_{ij} = \max_k \mathcal{P}_{ijk}$. Then, a linear layer and a sigmoid function are applied to map the pairwise features to the final prediction of edges, i.e., $S_{ij} = \text{Sigmoid}(\text{Linear}(\mathcal{S}_{ij}))$. Our learning label, the undirected graph representing the skeleton, can be easily calculated by summing the adjacency of the DAG G and its transpose G^T. Therefore, our learning target for the skeleton prediction task can be formulated as $\min \mathcal{L}(S, G + G^T)$, where $\mathcal{L}$ is the popularly used binary cross-entropy loss function.

V-structure Prediction. A UT $\langle X_i, X_k, X_j \rangle$ is a v-structure when $\exists \mathbf{S}$, such that $X_k \notin \mathbf{S}$ and $X_i \perp X_j | \mathbf{S}$. Motivated by this, we concatenate the corresponding pairwise features of the pair $\langle X_i, X_j \rangle$ with the node features of X_k as the feature for each UT $\langle X_i, X_k, X_j \rangle$ after a max-pooling along the observation dimension, i.e., $\mathcal{U}_{kij} = [\max_l \mathcal{P}_{ijl}; \max_l \mathcal{F}_{kl}]$. After that, we use a three-layer MLP with a sigmoid function to predict the existence of v-structures among all UTs, i.e., $\mathcal{U}_{kij} = \text{Sigmoid}(\text{MLP}(\mathcal{U}_{kij}))$. Given a data sample of d nodes, it outputs a third-order tensor of shape $\mathbb{R}^{d \times d \times d}$, namely v-tensor, corresponding to the predictions of the existence of v-structures. The v-tensor label can be obtained by $\mathcal{V}_{kij} = G_{ik} G_{jk} (1 - G_{ij})(1 - G_{ji})$, where $\mathcal{V}_{kij}$ indicates the existence of v-structure $X_i \rightarrow X_k \leftarrow X_j$. Therefore, the learning target for the v-structure prediction task can be formulated as $\min \mathcal{L}_{UT}(\mathcal{U}, \mathcal{V})$, where $\mathcal{L}_{UT}$ is the binary cross-entropy loss masked by UTs, i.e., we only calculate such loss on the valid UTs. In our current implementation, the parameters of the feature encoders are fine-tuned from the skeleton prediction task, as the UTs to be classified are obtained from the predicted skeleton and the skeleton prediction can be seen as a general pre-trained task.

Note that neural networks with our learning targets have a theoretical guarantee for correctness in asymptotic sense, as mentioned around the end of Sec. 3.

4.4 Post-Processing

Although our method theoretically guarantees asymptotic correctness, conflicts in predicted v-structures might occasionally occur in practice. Therefore, in the post-processing stage, we apply a straightforward heuristic to resolve the potential conflicts and cycles among predicted v-structures following previous work (Dai et al., 2023). After that, we use an improved version of Meek rules (Meek, 1995a; Tsagris, 2019) to obtain other MEC-identifiable edges without introducing extra cycles. Combining the skeleton from the skeleton predictor model with all MEC-identifiable edge directions, we get the CPDAG predictions.

We provide a more detailed description of the post-processing process in Appendix Sec. A3. It is worth noting that our current design of post-processing is a very conservative one, and this module is also non-essential in our whole framework; see Appendix Sec. A3 for more discussions and evidences.

5 EXPERIMENTS

In this section, we report the performance of SiCL on both synthetic and real-world benchmarks, followed by an ablation study. More results and discussions about time cost, generality, and acyclicity are deferred to Appendix Sec. A8, due to page limit.

5.1 Experiment Design

Metrics. We profile a causal discovery method's performance using the following two tasks:

Skeleton Prediction: Given a data sample D of d variables, for each variable pair, we want to infer if there exists direct causation between them. The standard metric **s-F1** (short for **skeleton-F1**) is used, which considers skeleton prediction as a binary classification task over the $d(d-1)/2$ variable pairs. For completion, we also report classification accuracy results. For methods with probabilistic outputs, AUC and AUPRC scores are also measured.

CPDAG Prediction: Given a data sample D of d variables, for each pair of variables, we aim to determine if there is direct causality between them. If so, we then assess whether the causal direction is MEC-identifiable, and if it is, we attempt to infer the specific causal direction. As this task involves both directed and undirected edge prediction, we use **SHD** (Structural Hamming Distance) to measure the difference between the true CPDAG and the inferred CPDAG. Besides that, we also measure the **o-F1** (short for **orientation-F1**) of the directed sub-graph of the inferred CPDAG (compared against the directed subgraph of the true CPDAG), which focuses on capturing the inference method's orientation capability in identifying *MEC-identifiable* causal edges. Finally, we calculate the **v-F1** score, where the F1 score is based on the set of v-structures in the true CPDAG as the positive instances (from all ordered triples), and the v-structures in the inferred CPDAG as the positive predictions.

Testing Data. A testing instance consists of a groundtruth causal graph G, the structural equations f_i and noise variables ϵ_i for each variable X_i, and an i.i.d. sample D. We use two categories of testing instances in our experiments:

Analytical Instances: where G is sampled from a DAG distribution $\mathcal{G}$, and $\{f_i, \epsilon_i\}$ sampled from a structural-equation distribution $\mathcal{F}$ and a noise meta-distribution $\mathcal{N}$. We consider three random graph distributions for $\mathcal{G}$: Watts-Strogatz (WS), Stochastic Block Model (SBM), Erdos-Rényi (ER); and three $\mathcal{F}$'s: random linear (L), Random Fourier Features (RFF), and conditional probability table (CPT). $\mathcal{N}$ is a uniform distribution over Gaussian's for continuous data, and a Dirichlet distribution over Multinomial Categorical distributions for discrete data. We examine five combinations of testing instances: **WS-L-G**, **SBM-L-G**, **WS-RFF-G**, **SBM-RFF-G**, and **ER-CPT-MC**.

Real-world Instance: The classic dataset **Sachs** is used to evaluate performance in real-world scenarios. It consists of a data sample recording the concentration levels of 11 phosphorylated proteins in 853 human immune system cells, and of a causal graph over these 11 variables identified by Sachs et al. (2005) based on expert consensus and biology literature.

Algorithms. As baselines, we compare with a series of representative unsupervised methods, including **PC** (using the recent Parallel-PC variation by Le et al. (2016)), **GES** (Chickering, 2002), **NOTEARS** (Zheng et al., 2018), **GOLEM** (Ng et al., 2020), **DAG-GNN** (Yu et al., 2019), **GRANDAG** (Lachapelle et al., 2020), **SLdisco** (Petersen et al., 2023) as well as **AVICI** (Lorch et al., 2022), a DNN-based SCL method regarded as current state-of-the-art method.

For our method, besides the full **SiCL** implementa-

Table 1: **General comparison of SiCL and other methods**. The average performance results in three runs are reported for SiCL. GES takes more than 24 hours per graph on WS-L-G. SLdicso is unsuitable on non-linear-Gaussian data. Full results on all metrics are provided in Appendix Tab. A5.

Method	WS-L-G		SBM-L-G		WS-RFF-G		SBM-RFF-G		ER-CPT-MC	
	s-F1↑	o-F1↑	s-F1↑	o-F1↑	s-F1↑	o-F1↑	s-F1↑	o-F1↑	s-F1↑	o-F1↑
PC	30.4	16.0	58.8	35.9	36.1	16.1	57.5	34.2	82.2	40.6
GES	*	*	70.8	55.0	41.7	23.6	56.5	38.0	82.1	42.4
NOTEARS	33.3	31.5	80.1	77.8	37.7	33.4	55.6	48.5	16.7	0.6
DAG-GNN	35.5	32.7	66.2	62.5	33.2	28.9	47.1	40.6	24.8	3.7
GRAN-DAG	16.6	11.7	22.6	14.4	4.7	1.1	17.4	3.8	40.8	7.3
GOLEM	30.0	19.3	68.5	65.2	27.6	17.7	41.1	24.8	37.6	9.3
SLdisco	0.1	0.1	1.9	1.2	*	*	*	*	*	*
AVICI	39.9	35.8	84.3	81.6	47.7	45.2	76.6	72.7	76.9	57.6
SiCL	**44.7**	**38.5**	**85.8**	**82.7**	**51.8**	**46.3**	**82.1**	**78.0**	**84.2**	**59.9**

tion as described by Sec. 4, we also implement **SiCL-Node-Edge**, which predicts the causal graph using the node features and can be regarded as equivalent to AVICI, and **SiCL-no-PF**, which skips pairwise feature extraction and predicts the skeleton and v-tensor using node-wise features (see Appendix Fig. A5). Notably, SiCL contains 2.8M parameters, while SiCL-Node-Edge and SiCL-no-PF contain 3.2M parameters, because SiCL contains fewer layers on the node feature encoder to eliminate potential bias from size difference.

For DNN-based SCL methods, the DNNs are trained with synthetic data where the causal graphs follow the Erdos-Rényi (ER) and Scale-Free (SF) models and the structural equations and noise variables follow the same distribution type as the corresponding testing data. Therefore, the disparities between the causal graph distribution at training and testing time help to examine the generality of SiCL in **OOD** settings to some extent. More details of the experimental setting are presented in Appendix Sec. A7.

5.2 Results on Synthetic Dataset

We conduct a comprehensive comparison of SiCL with various baselines in both skeleton prediction and CPDAG prediction tasks. The main results of metrics skeleton-F1 and orientation-F1 are presented in Tab. 1, and results on full metrics are provided in Appendix Tab. A5. On continuous data, DNN-based SCL methods (i.e., AVICI and SiCL) demonstrate consistent and obvious advantages over traditional approaches. SiCL consistently outperforms the other methods on both skeleton prediction task and CPDAG prediction task. On the other hand, some unsupervised methods achieve comparable performance among DNN-based SCL methods on the discrete data ER-CPT-MC. Nonetheless, our proposed SiCL emerges as the top performer, further substantiating its superiority in addressing the causal learning problem.

Table 2: Comparison on Sachs dataset.

Method	Skeleton Prediction		CPDAG Prediction	
	s-F1↑	s-Acc.↑	SHD↓	#v-struc.↓
PC	68.6	80.0	19	12
GES	70.6	81.8	19	8
DAG-GNN	21.1	72.7	15	**0**
NOTEARS	11.1	70.9	16	**0**
GRAN-DAG	45.5	78.2	12	**0**
GOLEM	36.4	74.5	14	**0**
AVICI	66.7	83.5	18	14
SiCL	**71.4**	**86.8**	**6**	**0**

5.3 Results on Real-world Dataset

To assess the practical applicability of SiCL, we conduct a comparison using the real-world dataset Sachs. The discretized Sachs data obtained from the bnlearn library[2] is used. The DNN-based SCL methods are trained on random synthetic graphs, making this also an **OOD** prediction task. The results are provided in Tab. 2.

For the skeleton prediction task, SiCL performs the best, albeit with a modest gap (generally 1~3 scores higher than the runners-up). For the CPDAG prediction task, SiCL performs significantly better than all other methods (reducing SHD from 12 to 6, against the second best). Interestingly, the true causal DAG of the Sachs benchmark actually contains no v-structure, so any predicted v-structure is an error. We see that methods competitive with SiCL in skeleton prediction (AVICI, PC, GES) mistakenly predicted a large number of v-structures on the Sachs data, while SiCL correctly predict zero v-structure.

[2]https://www.bnlearn.com/

Table 3: Ablation study of SiCL components. Full metrics are available in Appendix Tab. A6.

Method	WS-L-G		SBM-L-G	
	s-F1↑	o-F1↑	s-F1↑	o-F1↑
SiCL-Node-Edge	39.9	35.8	84.3	81.6
SiCL-no-PF	42.4	37.9	85.5	82.2
SiCL	**44.7**	**38.5**	**85.8**	**82.7**

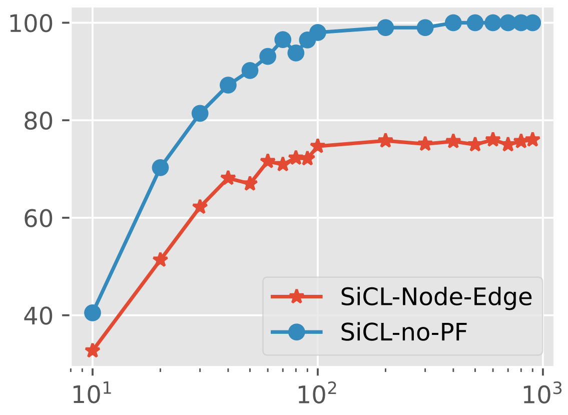


Figure 2: Comparison of SiCL-Node-Edge and SiCL-no-PF in o-F1 trend as observation samples increase on a constructed dataset.

5.4 Ablation Study

Effectiveness of Learning Identifiable Structures. As discussed in Section 4.3, SiCL focuses on learning MEC-identifiable causal structures rather than directly learning the adjacency matrix. To verify the effectiveness of this idea, we compare SiCL-Node-Edge with SiCL-no-PF. These two models share a similar node feature encoder architecture but have different learning targets: SiCL-Node-Edge predicts the adjacency matrix, while the SiCL-no-PF predicts the skeleton and v-tensor. The results are shown in Table 3. Consistently, SiCL-no-PF demonstrates higher performance on both skeleton and CPDAG prediction tasks. This observation echoes our theoretical conclusion regarding the necessity and benefits of learning identifiable causal structures to improve overall performance.

To further underscore the significance of learning identifiable causal structures (especially in the asymptomatic sense), we conduct a comparative analysis using a specially constructed dataset, which contains six nodes forming an independent v-structure and a UT. Figure 2 illustrates that the orientation F1 scores of CPDAG predictions from SiCL-Node-Edge suffer from an unavoidable error and do not improve with the addition of more observational samples. In contrast, predictions from SiCL-no-PF reach perfect accuracy, confirming the value of learning identifiable causal structures.

Effectiveness of Pairwise Representation. To assess the effectiveness of pairwise representation, we compare the full version of SiCL with a variant lacking pairwise features (SiCL-no-PF). As shown in Table 3, the full-version SiCL consistently outperforms SiCL-no-PF in both skeleton prediction and CPDAG prediction tasks. Notably, we have intentionally set the model size of the full-version SiCL (2.8M parameters) to be smaller than that of SiCL-no-PF (3.2M parameters) so as to avoid any potential advantage from increased model complexity brought by the pairwise feature encoder module. The observed performance gains in this case underscore the critical role of pairwise features in identifying causal structures. Additionally, we conduct further comparisons across more diverse settings, with results detailed in Appendix Sec. A7. These results demonstrate even more pronounced improvements in favor of SiCL, reinforcing the importance of pairwise representations in causal discovery.

6 CONCLUSION

We proposed SiCL, a novel DNN-based SCL approach designed to predict the corresponding skeleton and a set of v-structures. We showed that such design do not suffer from the (non-)identifiability limit that exists in current architectures. Moreover, SiCL is equipped with a pairwise encoder module to explicitly model relationships between node-pairs. Experimental results validated the effectiveness of these ideas.

This paper also introduces a few interesting open problems. The proposed DNN model works in the canonical setting under the classic MEC theory, in which the skeleton and v-structures are the identifiable structure. It can be an interesting future-work direction to explore how to learn other identifiable causal structure in other assumption settings following the same principle. Due to the inherent complexity of DNNs, the explanation of the decision mechanism of our model remains an open question. Therefore, future work could consider to explore how decisions are made within the networks and provide some insights for traditional methods. Moreover, the proposed pairwise encoder modules needs $O(d^3)$ computational complexity, which may restrict its current application to scenarios with huge number of nodes. Future work could focus on simplifying these operations or exploring features with less complexity (e,g., low rank features) to reduce the overall computational cost.

References

Andersson, S. A., Madigan, D., and Perlman, M. D. (1997). A Characterization of Markov Equivalence Classes for Acyclic Digraphs. *The Annals of Statistics*, 25(2):505–541.

Charpentier, B., Kibler, S., and Günnemann, S. (2022). Differentiable DAG Sampling. In *ICLR*.

Chickering, D. M. (2002). Optimal Structure Identification with Greedy Search. *Journal of Machine Learning Research*, 3(3):507–554.

Colombo, D., Maathuis, M. H., et al. (2014). Order-Independent Constraint-Based Causal Structure Learning. *Journal of Machine Learning Research*, 15(1):3741–3782.

Cussens, J. (2011). Bayesian Network Learning With Cutting Planes. In *UAI*.

Dai, H., Ding, R., Jiang, Y., Han, S., and Zhang, D. (2023). ML4C: Seeing Causality through Latent Vicinity. In *SDM*.

Frydenberg, M. (1990). The Chain Graph Markov Property. *Scandinavian Journal of Statistics*, 17(4):333–353.

Geffner, T., Antoran, J., Foster, A., Gong, W., Ma, C., Kiciman, E., Sharma, A., Lamb, A., Kukla, M., Pawlowski, N., Allamanis, M., and Zhang, C. (2022). Deep End-to-end Causal Inference. In *NeurIPS 2022 Workshop on Causality for Real-world Impact*.

Glymour, C., Zhang, K., and Spirtes, P. (2019). Review of Causal Discovery Methods Based on Graphical Models. *Frontiers in Genetics*, 10:524.

Hoyer, P., Janzing, D., Mooij, J. M., Peters, J., and Schölkopf, B. (2008). Nonlinear Causal Discovery with Additive Noise Models. In *NeurIPS*.

Immer, A., Schultheiss, C., Vogt, J. E., Schölkopf, B., Bühlmann, P., and Marx, A. (2023). On the Identifiability and Estimation of Causal Location-Scale Noise Models. In *ICML*.

Janzing, D., Mooij, J., Zhang, K., Lemeire, J., Zscheischler, J., Daniušis, P., Steudel, B., and Schölkopf, B. (2012). Information-Geometric Approach to Inferring Causal Directions. *Artificial Intelligence*, 182:1–31.

Ke, N. R., Chiappa, S., Wang, J. X., Bornschein, J., Goyal, A., Rey, M., Weber, T., Botvinick, M., Mozer, M. C., and Rezende, D. J. (2023). Learning to Induce Causal Structure. In *ICLR*.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT press.

Lachapelle, S., Brouillard, P., Deleu, T., and Lacoste-Julien, S. (2020). Gradient-Based Neural DAG Learning. In *ICLR*.

Lauritzen, S. L. (1996). *Graphical Models*, volume 17. Clarendon Press.

Le, T. D., Hoang, T., Li, J., Liu, L., Liu, H., and Hu, S. (2016). A Fast PC Algorithm for High Dimensional Causal Discovery With Multi-Core PCs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(5):1483–1495.

Lippe, P., Cohen, T., and Gavves, E. (2022). Efficient neural causal discovery without acyclicity constraints. In *ICLR*.

Lopez-Paz, D., Muandet, K., and Recht, B. (2015). The Randomized Causation Coefficient. *Journal of Machine Learning Research*, 16(90):2901–2907.

Lopez-Paz, D., Nishihara, R., Chintala, S., Scholkopf, B., and Bottou, L. (2017). Discovering Causal Signals in Images. In *CVPR*.

Lorch, L., Sussex, S., Rothfuss, J., Krause, A., and Schölkopf, B. (2022). Amortized Inference for Causal Structure Learning. In *NeurIPS*.

Ma, P., Ding, R., Dai, H., Jiang, Y., Wang, S., Han, S., and Zhang, D. (2022). ML4S: Learning Causal Skeleton from Vicinal Graphs. In *KDD*.

Meek, C. (1995a). Causal Inference and Causal Explanation with Background Knowledge. In *UAI*, pages 403–410.

Meek, C. (1995b). Strong Completeness and Faithfulness in Bayesian Networks. In *UAI*.

Ng, I., Ghassami, A., and Zhang, K. (2020). On the Role of Sparsity and DAG Constraints for Learning Linear DAGs. In *NeurIPS*, pages 17943–17954.

Pearl, J. (2009). *Causality*. Cambridge university press.

Peters, J., Mooij, J. M., Janzing, D., and Schölkopf, B. (2014). Causal Discovery With Continuous Additive Noise Models. *Journal of Machine Learning Research*, 15(1):2009–2053.

Petersen, A. H., Ramsey, J., Ekstrøm, C. T., and Spirtes, P. (2023). Causal Discovery for Observational Sciences Using Supervised Machine Learning. *Journal of Data Science*, 21(2):255–280.

Ramsey, J., Zhang, J., and Spirtes, P. L. (2012). Adjacency-Faithfulness and Conservative Causal Inference. *arXiv preprint arXiv:1206.6843*.

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.

Shimizu, S., Inazumi, T., Sogawa, Y., Hyvarinen, A., Kawahara, Y., Washio, T., Hoyer, P. O., Bollen, K., and Hoyer, P. (2011). DirectLiNGAM: A Direct Method for Learning a Linear Non-Gaussian Structural Equation Model. *Journal of Machine Learning Research*, 12(33):1225–1248.

Spirtes, P., Glymour, C., and Scheines, R. (2001). *Causation, Prediction, and Search.* MIT press.

Spirtes, P., Glymour, C. N., Scheines, R., and Heckerman, D. (2000). *Causation, prediction, and search.* MIT press.

Tsagris, M. (2019). Bayesian Network Learning with the PC Algorithm: An Improved and Correct Variation. *Applied Artificial Intelligence*, 33(2):101–123.

Varambally, S., Ma, Y., and Yu, R. (2024). Discovering Mixtures of Structural Causal Models from Time Series Data. In *ICML*.

Verma, T. and Pearl, J. (1990). Equivalence and Synthesis of Causal Models. In *UAI*.

Vowels, M. J., Camgoz, N. C., and Bowden, R. (2022). D'Ya Like Dags? A Survey on Structure Learning and Causal Discovery. *ACM Computing Surveys*, 55(4):1–36.

Weinberger, N. (2018). Faithfulness, Coordination and Causal Coincidences. *Erkenntnis*, 83(2):113–133.

Yu, K., Li, J., and Liu, L. (2016). A Review on Algorithms for Constraint-based Causal Discovery. *arXiv preprint arXiv:1611.03977*.

Yu, Y., Chen, J., Gao, T., and Yu, M. (2019). DAG-GNN: DAG Structure Learning with Graph Neural Networks. In *ICML*.

Zhang, K. and Hyvärinen, A. (2009). On the Identifiability of the Post-Nonlinear Causal Model. In *UAI*.

Zhang, K., Zhu, S., Kalander, M., Ng, I., Ye, J., Chen, Z., and Pan, L. (2021). gCastle: A Python Toolbox for Causal Discovery. *arXiv preprint arXiv:2111.15155*.

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. (2018). Dags With No Tears: Continuous Optimization for Structure Learning. In *NeurIPS*.

Zheng, Y., Huang, B., Chen, W., Ramsey, J., Gong, M., Cai, R., Shimizu, S., Spirtes, P., and Zhang, K. (2024). Causal-learn: Causal discovery in python. *Journal of Machine Learning Research*, 25(60):1–8.

Zhu, S., Ng, I., and Chen, Z. (2020). Causal Discovery with Reinforcement Learning. In *ICLR*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Yes]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Algorithm A1 SiCL Workflow for Predicting Causal Structures

```
Procedure INFERENCE(data, target)
Calculate node features with node encoder
Calculate pairwise features with pairwise encoder following Sec. 4.2
if target is skeleton then
    Calculate skeleton with Sec. 4.3
else
    Calculate v-structures with Sec. 4.3
end if
End Procedure

Procedure TRAINING_PHASE()
skeleton_predictor ← init_skeleton_predictor()
Sample graphs and corresponding data
Training the skeleton predictor with INFERENCE(data, skeleton)
Training the v-structure predictor with INFERENCE(data, v-structure), with feature encoders fine-tuned from
skeleton predictor
End Procedure

Procedure TESTING_PHASE(test_data)
Calculate predicted skeleton with the trained skeleton predictor
Calculate predicted v-structures with the trained v-structure predictor
Combine predicted skeleton and v-structures to obtain predicted CPDAG
End Procedure
```

A1 Theoretical Guarantee

In this section, we delve into the theoretical analysis concerning the asymptotic correctness of our proposed model with respect to the sample size. Sec. A1.1 lays out the essential definitions and assumptions pertinent to the problem under study. Following this, from Sec. A1.2 to A1.3, we rigorously demonstrate the asymptotic correctness of the neural network model. Finally, in Sec. A1.4, we engage in a detailed discussion about the practical advantages and superiority of neural network models.

A1.1 Definitions and Assumptions

As outlined in Sec. 2, a Causal Graphical Model is defined by a joint probability distribution P over d random variables $X_1, X_2, \cdots, X_d$, and a DAG G with d vertices representing the d variables. An observational dataset D consists of n records and d columns, which represents n instances drawn i.i.d. from P. In this work, we assume causal sufficiency:

Assumption A1.1 (Causal Sufficiency)**.** *There are no latent common causes of any of the variables in the graph.*

Moreover, we assume the data distribution P is Markovian to the DAG G:

Assumption A1.2 (Markov Factorization Property)**.** *Given a joint probability distribution P and a DAG G, P is said to satisfy Markov factorization property w.r.t. G if $P := P(X_1, X_2, \cdots, X_d) = \prod_{i=1}^{d} P\left(X_i \mid \mathrm{pa}_i^G\right)$, where pa_i^G is the parent set of X_i in G.*

It is noteworthy that the Markov factorization property is equivalent to the Global Markov Property (GMP) (Lauritzen, 1996), which is

Definition A1.1 (Global Markov Property (GMP))**.** *P is said to satisfy GMP (or Markovian) w.r.t. a DAG G if $X \perp_G Y|Z \Rightarrow X \perp Y|Z$. Here $\perp_G$ denotes d-separation, and $\perp$ denotes statistical independence.*

GMP indicates that any d-separation in graph G implies conditional independence in distribution P. We further assume that P is faithful to G by:

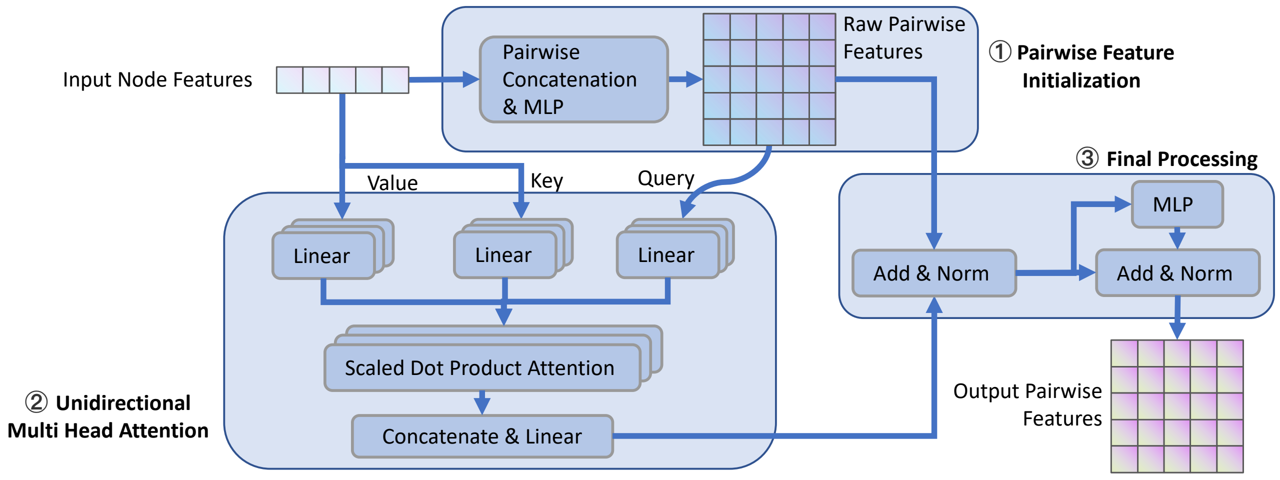


Figure A3: Illustration of the pairwise encoder module. In Part ①, it initializes raw pairwise features. In Part ②, a unidirectional attention is applied to utilized information from node features and pairwise features. In Part ③, an MLP and residual connection is used to yield final pairwise features.

Assumption A1.3 (Faithfulness)**.** *Distribution P is faithful w.r.t. a DAG G if $X \perp Y \,|\, Z \Rightarrow X \perp_G Y \,|\, Z$.*

Definition A1.2 (Canonical Assumption)**.** *We say our settings satisfy the canonical assumption if the Assumptions A1.1 - A1.3 are all satisfied.*

We restate the definitions of skeletons, Unshielded Triples (UTs) and v-strucutres as follows.

Definition A1.3 (Skeleton)**.** *A skeleton E defined over the data distribution P is an undirected graph where an edge exists between X_i and X_j if and only if X_i and X_j are always dependent in P, i.e., $\forall Z \subseteq \{X_1, X_2, \cdots, X_d\} \setminus \{X_i, X_j\}$, we have $X_i \not\perp X_j | Z$.*

Under our assumptions, the skeleton is the same as the corresponding undirected graph of G (Spirtes et al., 2000).

Definition A1.4 (Unshielded Triples (UTs) and V-structures)**.** *A triple of variables X, T, Y is an Unshielded Triple (UT) denoted as $\langle X, T, Y \rangle$, if X and Y are both adjacent to T but not adjacent to each other in the DAG G or the corresponding skeleton. It becomes a v-structure denoted as $X \to T \leftarrow Y$, if the directions of the edges are from X and Y to T in G.*

We introduce the definition of separation set as:

Definition A1.5 (Separation Set)**.** *For a node pair X_i and X_j, a node set Z is a separation set if $X_i \perp X_j | Z$. Under faithfulness assumption, a separation set Z is a subset of variables within the vicinity that d-separates X_i and X_j.*

Finally, we assume a neural network can be used as a universal approximator in our settings.

Assumption A1.4 (Universal Approximation Capability)**.** *A neural network model can be trained to approximate a function under our settings with arbitary accuracy.*

A1.2 Skeleton Learning

In this section, we prove the asymptotic correctness of neural networks on the skeleton prediction task by constructing a perfect model and then approximating it with neural networks. For the sake of convenience and brevity in description, we define the skeleton predictor as follows.

Definition A1.6 (Skeleton Predictor)**.** *Given observational data D, a skeleton predictor is a predicate function with domain as observational data D and predicts the adjacency between each pair of the vertices.*

Now we restate the Remark from Ma et al. (2022) as the following proposition. It proves the existence of a perfect skeleton predictor by viewing the skeleton prediction step of PC (Spirtes et al., 2000) as a skeleton predictor, which is proved to be sound and complete.

Proposition A1.1 (Existence of a Perfect Skeleton Predictor)**.** *There exists a skeleton predictor that always yields the correct skeleton with sufficient samples in D.*

Proof. We construct a skeleton predictor SP consisting of two parts by viewing PC (Spirtes et al., 2000) as a skeleton predictor. In the first part, it extracts a pairwise feature $\boldsymbol{x}_{ij}$ for each pair of nodes X_i and X_j:

$$\boldsymbol{x}_{ij} = \min_{Z \subseteq V \setminus \{X_i, X_j\}} \left\{X_i \sim X_j \mid Z\right\}, \tag{2}$$

where $\left\{X_i \sim X_j \mid Z\right\} \in [0,1]$ is a scalar value that measures the conditional dependency between X_i and X_j given a node subset Z. Consequently, $\boldsymbol{x}_{ij} > 0$ indicates the persistent dependency between the two nodes.

In the second part, it predicts the adjacency based on $\boldsymbol{x}_{ij}$:

$$\left(X_i, X_j\right) = \begin{cases} 1 \text{ (adjacent)} & \boldsymbol{x}_{ij} \neq 0 \\ 0 \text{ (non-adjacent)} & \boldsymbol{x}_{ij} = 0 \end{cases} \tag{3}$$

Now we prove that SP always yields the correct skeleton by proving the absence of false positive predictions and false negative predictions. Here, false positive prediction denotes SP predicts a non-adjacent node pair as adjacent and false negative predictions denote SP predicts an adjacent node pair as non-adjacent.

- **False Positive.** Suppose X_i, X_j are non-adjacent. Under the Markovian assumption, there exists a set of nodes Z such that $\left\{X_i \sim X_j \mid Z\right\} = 0$ and hence $\boldsymbol{x}_{ij} = 0$. According to Eq. (3), SP will always predicts them as non-adjacent.
- **False Negative**. Suppose X_i, X_j are adjacent. Under the faithfulness assumption, for any $Z \in V \setminus \left\{X_i, X_j\right\}, \left\{X_i \sim X_j \mid Z\right\} > 0$, which implies $\boldsymbol{x}_{ij} > 0$. Therefore, SP always predicts them as adjacent.

Therefore, SP never yields any false positive predictions or false negative predictions under the Markovian assumption and faithfulness assumption, i.e., it always yields the correct skeleton. □

With the existence of a perfect skeleton predictor, we prove the correctness of neural network models with sufficient samples under our assumptions.

Theorem A1.1. *Under the canonical assumption and the assumption that neural network can be used as a universal approximator (Assumption A1.4), there exists a neural network model that always predicts the correct skeleton with sufficient samples in D.*

Proof. From Proposition A1.1, there exists a perfect skeleton predictor that predicts the correct skeleton. Thus, according to the Assumption A1.4, a neural network model can be trained to approximate the perfect skeleton prediction hence predicts the correct skeleton. □

A1.3 Orientation Learning

Similarly to the overall thought process in Sec. A1.2, in this section we prove the asymptotic correctness of neural networks on the v-structure prediction task by constructing a perfect model and then approximating it with neural networks.

Definition A1.7 (V-structure Predictor)**.** *Given observational data D with sufficient samples from a BN with vertices $V = \{X_1, \ldots, X_p\}$, a v-structure predictor is a predicate function with domain as observational data D and predicts existence of the v-structure for each unshielded triple.*

The following proposition proves the existence of a perfect v-structure predictor by viewing the orientation step of PC (Spirtes et al., 2000) as a v-structure predictor.

Proposition A1.2 (Existence of a Perfect V-structure Predictor)**.** *Under the Markov assumption and faithfulness assumption, there exists skeleton predictor that always yields the correct skeleton.*

Proof. We construct a v-structure predictor VP consisting of two parts by viewing PC (Spirtes et al., 2000) as a v-structure predictor. In the first part, it extracts a boolean feature $\boldsymbol{z}_{kij}$ for each UT $\langle X_i, X_k, X_j \rangle$:

$$\boldsymbol{z}_{kij} = (X_k \in Z), \text{ where } Z \text{ is called as a sepset, i.e. } X_i \perp Y_j | Z. \tag{4}$$

Note that the sepset Z always exists because the separation set of a UT always exists (See Lemma 4.1 in Dai et al. (2023)).

In the second part, it predicts the v-structures based on $\boldsymbol{z}_{ijk}$:

$$\langle X_i, X_k, X_j \rangle = \begin{cases} 0 \text{ (not v-structure)} & \boldsymbol{z}_{kij} = True \\ 1 \text{ (v-structure)} & \boldsymbol{z}_{kij} = False \end{cases} \tag{5}$$

Now we prove that VP always yields the correct predictions of v-structures. According to Theorem 5.1 on p.410 of Spirtes et al. (2000), assuming faithfulness and sufficient samples, if a UT $\langle X_i, X_k, X_j \rangle$ is a v-structure, then X_k does not belong to any separation sets of (X_i, X_j); if a UT $\langle X_i, X_k, X_j \rangle$ is not a v-structure, then X_k belongs to every separation sets of (X_i, X_j). Therefore, we have $\boldsymbol{z}_{kij} = False$ if and only if X_k is not in any separation set of X_i and X_j, i.e., $\langle X_i, X_k, X_j \rangle$ is a v-structure. □

With the existence of a perfect v-structure predictor, we prove the correctness of neural network models with sufficient samples under our assumptions.

Theorem A1.2. *Under the canonical assumption and the assumption that neural network can be used as a universal approximator (Assumption A1.4), there exists a neural network model that always predicts the correct v-structures with sufficient samples in D.*

Proof. From Proposition A1.1, there exists a perfect skeleton predictor that predicts the correct v-structures. Thus, according to the Assumption A1.4, a neural network model can be trained to approximate the perfect v-structure predictions hence predicts the correct v-structures. □

A1.4 Discussion

In the sections above, we prove the asymptotic correctness of neural network models by constructing theoretically perfect predictors. These predictors both consist of two parts: feature extractors providing features $\boldsymbol{x}_{ij}$ and $\boldsymbol{z}_{ijk}$, and final predictors of adjacency and v-structures. Even though they have a theoretical guarantee of the correctness with sufficient samples, it is noteworthy that they are hard to be applied practically. For example, to obtain $\boldsymbol{x}_{ij}$ in Eq. (2), we need to calculate the conditional dependency between X_i and X_j given every node subset $Z \subseteq V \setminus \{X_i, X_j\}$. Leaving aside the fact that the number of Zs itself presents factorial complexity, the main issue is that when Z is relatively large, due to the curse of dimensionality, it becomes challenging to find sufficient samples to calculate the conditional dependency. This difficulty significantly hampers the ability to apply the constructed prefect predictors in practical scenarios.

Some existing methods can be interpreted as constructing more practical predictors. Majority-PC (MPC) (Colombo et al., 2014) achieves better performance on finite samples by modifying Eq. (4) - (5) as:

$$\boldsymbol{z}_{kij} = \frac{\left|\{(X_k, Z) | \{X_i \sim X_j | Z\} = 0 \land X_k \in Z\}\right|}{\left|\{Z | \{X_i \sim X_j | Z\} = 0\}\right|}, \tag{6}$$

and

$$\langle X_i, X_k, X_j \rangle = \begin{cases} 0 \text{ (not v-structure)} & \boldsymbol{z}_{ijk} > 0.5 \\ 1 \text{(v-structure)} & \boldsymbol{z}_{ijk} \leq 0.5, \end{cases} \tag{7}$$

where $\{X_i \sim X_j \mid Z\} \in [0, 1]$ is a scalar value that measures the conditional dependency between X_i and X_j given a node subset Z, and $|\cdot|$ represents the cardinality of a set. Due to its more complex classification mechanism, it achieves better performance empirically. However, from the machine learning perspective, features from both the

Algorithm A2 Post-processing

```
Input: weighted skeleton matrix $S$, weighted V-tensor $U$, threshold for skeleton $\tau_s$, threshold for v-structure $\tau_v$
Output: predicted oriented edge set oriEdges, predicted skeleton skeleton
Step 1:
// Obtaining a predicted skeleton by thresholding.
skeleton = $\{(i,j) | max(S_{ij}, S_{ji}) > \tau_s\}$
// Obtaining raw v-structures $\texttt{vstructs}_{\texttt{raw}}$ by thresholding.
$\texttt{vstructs}_{\texttt{raw}} = \{(i,j,k) | (i,j) \in \texttt{skeleton}$ and $(i,k) \in \texttt{skeleton}$ and $(j,k) \notin \texttt{skeleton}$ and $max(U_{ijk}, U_{ikj}) > \tau_v\}$
Step 2:
// V-structure conflict resolving: discard any v-structure if there exists another conflicted v-structure with a higher predicted score, following (Dai et al., 2023).
$\texttt{vstructs} = \{(i,j,k) \in \texttt{vstructs}_{\texttt{raw}} | \forall (i',j',k') \in \texttt{vstructs}_{\texttt{raw}}, (i' \neq k$ and $i' \neq j)$ or $(k' \neq i$ and $j' \neq i)$ or $U_{i'j'k'} < U_{ijk}\}$
Step 3:
// Obtaining the predicted directed edge from vstructs.
$\texttt{oriEdges}_{\texttt{raw}} = \{(j,i) | \exists k, (i,j,k) \in \texttt{vstructs}\}$
//Set a score for each edge with the highest v-structure's score containing this edge.
Set $\{p_{ij}\}$ such that $p_{ij} = max_v U_v$ for $v \in \texttt{oriEdges}_{\texttt{raw}}$ and $v \ni (i,j)$.
// If there exist any cycles, remove the edge with the smallest score in each cycle.
$\texttt{oriEdges} = \{(i,j) \in \texttt{oriEdges}_{\texttt{raw}} | \forall$cycle $C, (i,j) \notin C$ or $(\exists (i',j') \in C, p_{ij} > p_{i'j'})\}$
Step 4:
// Meek rules: Add edges to oriEdges for directed edges that (1) introducing the edges does not lead to cycles or new v-structures; (2) adding the opposite edges necessarily leads to cycles or new v-structures.
$\texttt{oriEdges} = \texttt{oriEdges} \cup \{(i,j) \in \texttt{skeleton} | (i,j)$ complies with Meek rules$\}$
```

PC and MPC predictors are relatively simple. As supervised causal learning methods, ML4S (Ma et al., 2022) and ML4C (Dai et al., 2023) provide more systematic featurizations by manual feature engineering and utilization of powerful machine learning models for classification. While these methods show enhanced practical efficacy, their manual feature engineering processes are complex. In our paper, we utilize neural networks as universal approximators for learning the prediction of identifiable causal structures. It not only simplifies the procedure but also potentially uncovers more nuanced and complex patterns within the data that manual methods might overlook. It is noteworthy that the benefits of supervised causal learning using neural networks are also discussed elsewhere, as mentioned in SLdisco (Petersen et al., 2023) and CSIvA (Ke et al., 2023).

A2 More Discussions on Identifiability and Causal Assumptions

Advocation of Learning Identifiable Structures under All Settings. In this paper, we have to work on a concrete setting for demonstration purpose with concrete identifiable causal structures in this paper. Nonetheless, we want to emphasize that the very concept of identifiability, as well as its ramifications in SCL, is indeed a general issue that is less bound to the issue of "which causal structures are identifiable under which assumptions". The simple fact that in some situations the causal edge cannot be identified – no matter what feature can be identified in that case – this identifiability limit has a general effect on SCL. Unless the causal graph/edge itself becomes fully invariant/identifiable (a special case that is important but certainly not universally true), the presence of the identifiability limit entails a fundamental bias for a popular SCL model architecture (i.e., Node-Edge) that cannot be mitigated by larger model or bigger data at all. This "identifiability-limit-causes-learning-error" effect is the main thesis of this paper, and we advocate to design neural networks that focus on learning the identifiable features (no matter what those features are). In other words, there is nothing stopping one from studying another setting where another feature is identifiable though, and in that case we would also advocate to learn that feature instead of v-structures. For example, if we assume canonical MEC assumptions and non-existence of causal-fork and v-structures, the identifiable causal structure becomes a kind of chains. In that case, one may want to design neural networks that predict about causal chains.

Table A4: The o-F1 comparison between the used conflict resolving method with an opposite variant.

Conflict Resolving Method	WS-L-G	SBM-L-G
Original Conflict Resolving	**41.1**	**83.3**
Opposite Conflict Resolving	40.7	83.2

Rationality of Canonical Assumptions. In this paper, we choose the canonical setting under the classic MEC theory, in which the skeleton and v-structures are the identifiable structure. This setting includes the assumptions of the Markov and faithfulness conditions. Unlike scenario-specific assumptions, such as those tied to a particular data-generating process, these assumptions are classic assumptions about causality that are often adopted as "postulates" about some general aspects of the world. For example,

- Pearl (2009) argues that stability (faithfulness) stems from the natural improbability of strict equality constraints among parameters, which aligns with the autonomy of causal mechanisms.
- Spirtes et al. (2001) support the Causal Faithfulness Condition (CFC) by noting that the exact cancellation of causal paths is highly improbable under natural conditions.
- Weinberger (2018) reinforces this argument, proposing that coincidences leading to CFC violations are rare and lack explanatory power, further justifying its adoption within a general modeling framework.

These considerations underscore the rationality and generality of the assumptions, making them a natural choice for our analysis.

A3 Details and Discussion about Post-processing

For comprehensive clarity, we provide a clear process about the post-processing algorithm in Alg. A2.

Discussion. It is worth noting that our design in post-processing is as conservative as possible. In fact, we simply adhere to the conventions in deep learning (i.e., thresholding) to obtain the skeleton and the initial v-structure set. Subsequently, we follow the conventions in constraint-based causal discovery methods to derive the final oriented edges. Therefore, we have not dedicated extensive efforts towards the meticulous design, nor do we intend to emphasize this aspect of our workflow.

The conflicts and cycles are not unique to SiCL; they are, in fact, common issues encountered by all constraint-based algorithms like PC. Moreover, it's worth noting that they never appear if the networks perform perfect. Therefore, the conflict resolving of v-structures and the removal of cycles are designed as fallback mechanisms to ensure the soundness of our workflow, rather than being central elements of our approach. To illustrate it, we experimented with an opposite variant (Intuitively, this is a bad choice) that prioritizes discarding the v-structure with the higher predicted probability. The minimal differences in outcomes between this variant and its counterpart, as detailed in Tab. A4, support our viewpoint that the conflict resolution process is of limited significance within our workflow. On the other hand, experimental results presented in Tab. A11 underscore the infrequency of cycles in the predictions, reinforcing the non-essential nature of the cycle removal component.

A4 Details about Node Feature Encoder

Motivated by previous approaches (Lorch et al., 2022; Ke et al., 2023), we employ a transformer-like architecture comprising attention layers over either the observation dimension or the node dimension alternately as the node feature encoder. Concretely, for the raw node features $\mathcal{F} \in \mathbb{R}^{d \times n \times h}$ corresponding to d nodes and n observations, our goal is to capture the correlations between both different nodes and different observations. Therefore, we utilize two transformer encoder layers over the observation dimension and the node dimension alternatively:

$$\begin{aligned} &\mathcal{F} \leftarrow TransformerEncoderLayer(\mathcal{F},\mathcal{F},\mathcal{F}) \\ &\mathcal{F} \leftarrow \mathcal{F}.transpose(0,1) \\ &\mathcal{F} \leftarrow TransformerEncoderLayer(\mathcal{F},\mathcal{F},\mathcal{F}) \\ &\mathcal{F} \leftarrow \mathcal{F}.transpose(0,1). \end{aligned} \tag{8}$$

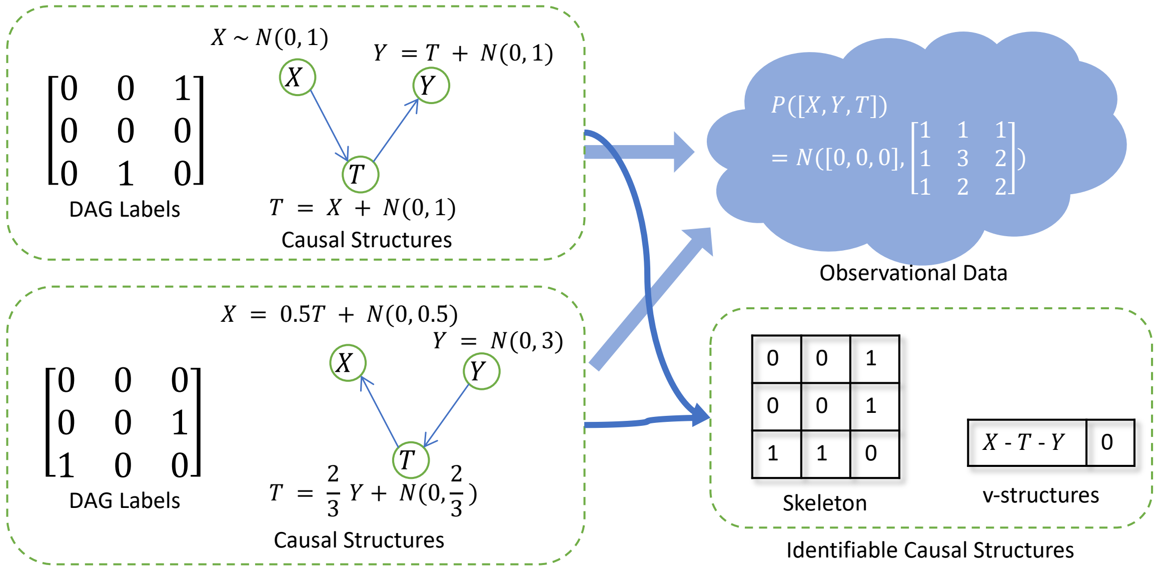
The above operation is repeated multiple times for sufficiently feature encoding. It yields the final node feature tensor $\mathcal{F} \in \mathcal{R}^{d \times \times h}$.

A5 Illustration of the Case Study in Sec. 3

Fig. A4 presents an illustration for the case study of the Node-Edge approach in Sec. 3. It clearly shows that observational data with the two different parametrized forms follow the same joint distribution:

$$P([X,Y,T]) = \mathcal{N}\left([0,0,0], \begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 2 \\ 1 & 2 & 2 \end{bmatrix}\right). \tag{9}$$

Therefore, the observational datasets coming from the two DAGs are inherently indistinguishable.



$X \sim N(0,1)$
$Y = T + N(0,1)$
$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$
DAG Labels
$T = X + N(0,1)$
Causal Structures
$P([X,Y,T])$
$= N([0,0,0], \begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 2 \\ 1 & 2 & 2 \end{bmatrix})$
Observational Data
$X = 0.5T + N(0,0.5)$
$Y = N(0,3)$
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$
DAG Labels
$T = \frac{2}{3}Y + N(0,\frac{2}{3})$
Causal Structures
0 0 1
0 0 1
1 1 0
Skeleton
X - T - Y 0
v-structures
Identifiable Causal Structures

Figure A4: The problem setting to emphasize the limitations of the Node-Edge approach. *Best viewed in color.*

A6 Proof and Discussion for Proposition 3.1

We first restate the Proposition 3.1 with more details and provide the proof.

Proposition A6.1. *Let $\mathcal{G}_n$ be the set of graphs with $n+1$ nodes where there is a central node y such that (1) every other node is connected to y, (2) there is no edge between the other nodes, (3) there is at most one edge pointing to y. For any distribution Q over $\mathcal{G}_n$, let $M(Q)$ be another distribution over $\mathcal{G}_n$ such that for any causal edges e, e', $P_{G\sim Q}(e \in G) = P_{G\sim M(Q)}(e \in G) = P_{G\sim M(Q)}(e \in G|e' \in G)$. We have*

$$\max_Q P_{G\sim M(Q)}(G \notin \mathcal{G}_n) = 1 - \frac{2n-1}{n-1}(1-\frac{1}{n})^n. \tag{10}$$

As a corollary, we have

$$\sup_n \max_Q P_{G\sim M(Q)}(G \notin \mathcal{G}_n) = 1 - \frac{2}{e} \approx 0.2642, \tag{11}$$

Proof. Denote other nodes except for the central node as x_i where $i \in \{1, 2, \ldots, n\}$. In our setting, the set $\mathcal{G}_n$ contains $n+1$ DAGs with the same skeleton and no v-structure : $G_0 : y \to x_i$ for all x_i, and $G_i : y \to x_j$ for all $x_j \neq x_i$ together with $x_i \to y$. Denote the sampling probability of DAG G_i from $\mathcal{G}_n$ as P_i. Therefore, the marginal probability of the edge $y \to x_i$ is $1 - P_i$.

If $\exists i$, $P_i = 1$, it means that $\mathcal{G}_n$ only contains the DAG G_i. Therefore, $M(Q)$ is equivalent to Q and we have $P_{G \sim M(Q)}(G \notin \mathcal{G}_n) = 0$.

If $\forall i$, $P_i < 1$, denoting $Q_i = 1 - P_i$ and $P(v)$ as the probability of G containing no v-structures. In other words, $P(v) = P_{G \sim M(Q)}(G \in \mathcal{G}_n)$. We have

$$P(v) = \prod_{i=1}^{n} Q_i + \sum_{j=1}^{n} \frac{\prod_i^n Q_i}{Q_j}(1 - Q_j) = (\prod_{i=1}^{n} Q_i) \cdot (1 + \sum_{j=1}^{n} \frac{1 - Q_j}{Q_j}). \tag{12}$$

As $P(v)$ is a probability, we have $P(v) > 0$. Denoting function

$$f(Q_1, Q_2, \ldots, Q_n) = \log P(v) = \sum_{i=1}^{n} \log Q_i + \log(1 + \sum_{j=1}^{n} \frac{1 - Q_j}{Q_j}), \tag{13}$$

we would like to find its minimum s.t. $\sum_i Q_i \geq n - 1$ and $Q_i \in (0, 1]$.

Define its Lagrange function

$$L(Q_1, Q_2, \ldots, Q_n, \lambda) = f + \lambda(n - 1 - \sum_i Q_i). \tag{14}$$

We have

$$\frac{\partial L}{\partial \lambda} = n - 1 - \sum_i Q_i, \tag{15}$$

and

$$\frac{\partial L}{\partial Q_i} = \frac{1}{Q_i}(1 - \frac{1}{Q_i(1 - n + \sum_{k=1}^{n} \frac{1}{Q_k})}) - \lambda. \tag{16}$$

Now we are going to find the extremums for $L(Q_1, Q_2, \ldots, Q_n, \lambda)$.

(1) If $\lambda = 0$, we have $\forall i$, $\frac{\partial f}{\partial Q_i} = 0$, then

$$\forall i, Q_i = \frac{1}{(1 - n + \sum_{k=1}^{n} \frac{1}{Q_k})}. \tag{17}$$

It indicates that $\forall i, Q_i = 1$, hence $f = 0$ and $P(v) = 1$.

(2) If $\lambda \neq 0$, $\exists i$, we have $\forall i$, $\frac{\partial f}{\partial Q_i} = \lambda$ and $\sum_{i=1}^{n} = n - 1$. In other words, we have

$$\forall i, j, \frac{\partial f}{\partial Q_i} = \frac{\partial f}{\partial Q_j} = \lambda. \tag{18}$$

Define function

$$h(Q_i) = \frac{\partial f}{\partial Q_i} = \frac{1}{Q_i}(1 - \frac{1}{Q_i(1 - n + \sum_{k=1}^{n} \frac{1}{Q_k})}). \tag{19}$$

we can rewrite the function as

$$h(Q_i) = \frac{1}{Q_i}(1 - \frac{1}{1 + AQ_i}), \tag{20}$$

where $A = 1 - n + \sum_{k \neq i} \frac{1}{Q_k} \geq 1 - n + \frac{(n-1)^2}{n-1-Q_i} > 0$. Therefore, $h(x)$ is a monotonic function in its domain.

It indicates that $\forall i, j$, $Q_i = Q_j = \frac{n-1}{n}$, where $P(v) = \frac{2n-1}{n-1}(1 - \frac{1}{n})^n$.

Now we are going to list the boundary points for f.

(1) $\forall i$, $Q_i = 1$, it becomes the first extremum point.

(2) $\exists i$, Q_i is approaching to 0. Due to the constraint of $\sum Q_i \geq n-1$, other Qs are approaching to 1. We have $\lim_{Q_i \to 0} f = 0$ and $P(v) = 1$.

In conclusion, the maximum point of function f is $\forall i$, $Q_i = \frac{n-1}{n}$, where

$$P(v) = \frac{2n-1}{n-1}(1 - \frac{1}{n})^n, \tag{21}$$

and

$$P_{G \sim M(Q)}(G \notin \mathcal{G}_n) = 1 - P(v) = 1 - \frac{2n-1}{n-1}(1 - \frac{1}{n})^n. \tag{22}$$

□

Discussion. It is worth noting that $\mathcal{G}_n$ is exactly the MEC of any graph in $\mathcal{G}_n$. Hence, $P_{G \sim M(Q)}(G \notin \mathcal{G}_n)$ represents the probability that the graph sampled from $M(Q)$ is incorrect. It indicates that a Node-Edge model could suffer from an inevitable error rate of 0.2642 though has been perfectly trained to predict $M(Q)$.

A7 Experimental Settings

Baselines. To demonstrate the effectiveness and superiority of the proposed framework, several representative baselines from multiple categories are selected for comparison. The PC algorithm is a classic constraint-based causal discovery algorithm based on conditional independence tests, and the version with parallelized optimization is selected (Le et al., 2016). GES, a classic score-based greedy equivalence search algorithm, is also included (Chickering, 2002). For continuous optimization methods, we compare with NOTEARS (Zheng et al., 2018), a representative gradient-based optimization method, and GOLEM (Ng et al., 2020), regarded as a more efficient variant of NOTEARS. For neural network based optimization algorithms, we compare with DAG-GNN (Yu et al., 2019), an optimization algorithm based on graph neural networks, and GRAN-DAG, a gradient-based algorithm using neural network modeling (Lachapelle et al., 2020). For DNN-based SCL methods, we compare with AVICI, which is the most related work to ours and regarded as the current state-of-the-art method (Lorch et al., 2022).

Implementation Details. The implementation from gCastle (Zhang et al., 2021) is utilized for baselines except the SCL methods (i.e., SLdisco and AVICI). For PC algorithm, we employ the Fisher-Z transformation with a significance threshold of 0.05 for conditional independence tests, which is a prevalent choice in statistical analyses and current PC implementations (Zhang et al., 2021; Zheng et al., 2024). Our criterion for graph selection in GES experiments is the Gaussian Bayesian Information Criterion (BIC), specifically the l_∞-penalized Gaussian likelihood score. It is used in the original paper (Chickering, 2002), and remains a favored variant in the literature. For NOTEARS, adhering to the official implementation's settings, we configure NOTEARS with a maximum of 100 dual ascent steps, and an edge dropping threshold of 0.3. For hyperparameters lacking specific default settings, such as the L1 penalty and loss function type, we default to settings used by gCastle Zhang et al. (2021), employing an L1 penalty of 0.1 and an L2 loss function. For DAG-GNN, we utilize hyperparameter settings directly from the original implementation, ensuring consistency with established benchmarks. For GOLEM and GRAN-DAG, we also use the default setting of gCastle (Zhang et al., 2021). Note that the CSIvA model (Ke et al., 2023) is also a closely related method, but it is not compared due to the unavailability of its relevant codes and its requirement for interventional data as input. The original implementation of SLdisco Petersen et al. (2023) was developed in R. To enhance compatibility with our data generation and evaluation workflows, we reimplemented the model using PyTorch. The original AVICI model (Lorch et al., 2022) does not support discrete data. Therefore, we use an embedding layer to replace its first linear layer when using AVICI on discrete data.

Synthetic Data. We randomly generate random graphs from multiple random graph models. For continuous data, following previous work (Lorch et al., 2022), Erdős-Rényi (ER) and Scale-free (SF) are utilized as the

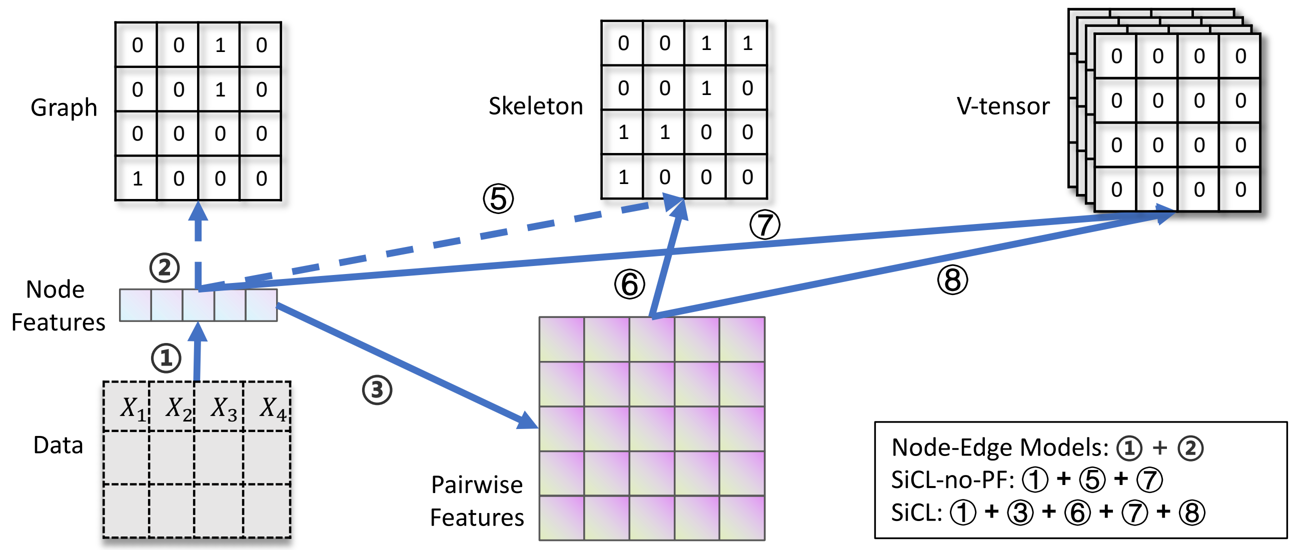


Figure A5: Illustration of the architecture comparison of Node-Edge models, SiCL-no-PF and SiCL.

training graph distribution $p(G)$. The degree of training graphs in our experiments varies randomly among 1, 2, and 3. For testing graph distributions, Watts-Strogatz (WS) and Stochastic Block Model (SBM) are used, with parameters consistent with those in the previous paper (Lorch et al., 2022). All synthetic graphs for continuous data contain 30 nodes. The lattice dimension of Watts-Strogatz (WS) graphs is sampled from $\{2, 3\}$, yielding an average degree of about 4.92. The average degrees of Stochastic Block Model (SBM) graphs are set at 2, following the settings in the aforementioned paper. For discrete data, 11-node graphs are used. SF is utilized as the training graph distribution $p(G)$ and ER is used for testing. The synthetic training data is generated in real-time, and the training process does not use the same data repeatedly. All synthetic test datasets contain 100 graphs, and the average values of the metrics on the 100 graphs are reported to comprehensively reflect the performance.

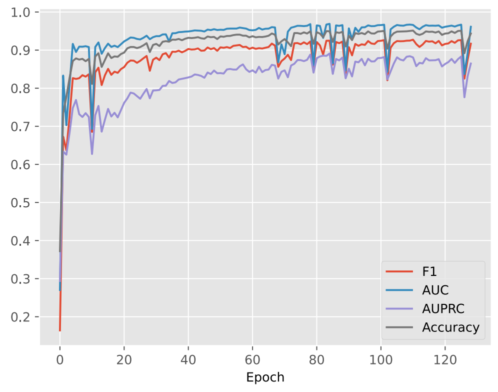
For the forward sampling process from graph to continuous data, both the linear Gaussian mechanism and general nonlinear mechanism are applied. Concretely, the Random Fourier Function mechanism is used for the general nonlinear data following the previous paper (Lorch et al., 2022). In synthesizing discrete datasets, the Bernoulli distribution is used following previous papers (Dai et al., 2023; Ma et al., 2022).

More Implementation Details and Computational Resources. The two network modules, i.e., the SPN and VPN, are optimized by Adam optimizer with default hyperparameters. Following previous work (Lorch et al., 2022), the training batch size is set as 20. All classic algorithms are run on an AMD EPYC 7V13 CPU, and DNN-based methods are run on Nvidia 1080Ti, A40 and A100 GPUs. Training SiCL on a 30-node training set with batch size 20 needs about 60GB memory, and training on a 11-node training set needs about 20GB memory. The learning rate is 3×10^{-4}. The batch size is 15, and the DNN models are trained for 1.2×10^{5} batches by default.

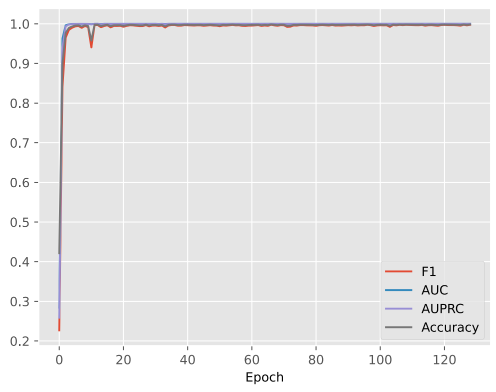
A8 Extra Experimental Results

A8.1 Effectiveness of V-structure Prediction Network

Fig. A6 illustrates the test performance trends of the v-structure prediction model on SBM and WS random graphs during the training process. In this model, the feature extractor FE is fine-tuned from the skeleton prediction model. The performance increases rapidly and achieves a relatively high level after just a few initial epochs. This suggests that our v-structure prediction network is capable to predict v-structures, and indicates that the pre-trained pairwise features from the skeleton prediction model are both effective and generalizable.



(a) WS-LG



(b) SBM-LG

Figure A6: Variation trends of the test performance of the V-structure Prediction Network on WS-LG and SBM-LG during training.

A8.2 More Evidence on Effectiveness of Pairwise Representation

To further support the effectiveness of using pairwise representation, we present additional experimental results on different training datasets and test datasets, including ER-L-G, SF-L-G, ER-RFF-G, and SF-RFF-G. For models, we compare SiCL with a variant without pairwise representation, i.e., SiCL-no-PF.

The results are provided in Tab. A7. Models with pairwise representation ourperform the corresponding baseline models under almost all comparisons, further verifying the effectiveness of using pairwise representation in models.

A8.3 Additional Comparison on DAG Prediction

We provide an additional comparison with the AVICI baseline on the DAG prediction task. Since SiCL predicts CPDAGs and does not directly produce DAG predictions, we corrected the DAG predictions from AVICI using the edge directions inferred from the CPDAGs predicted by SiCL. The results, summarized in the Table A8, demonstrate that incorporating CPDAG-inferred edge directions improves the DAG prediction metrics. This further confirms the effectiveness and generality of our approach, even in tasks focused on DAG metrics.

A8.4 Comparison with Autoregressive models on Inference Time Costs

To validate that the autoregressive models have a relatively high time costs due to the quadratic number of inference runs w.r.t. number of variables, we reproduce the network architecture of a representative autoregressive model, i.e., CSIvA (Ke et al., 2023), and compare SiCL with it. We use the same random input for both the models with increasing number of variables. The results are provided in Fig. A7. The time costs of the autoregressive model show a fast increasing trend and are much more than costs of SiCL, validating the correctness of our analysis.

A8.5 Training Data Diversity and Model Generalization

We present experimental evidence that highlights the significant contribution of training data diversity to the model's generalization capabilities, even when applied to out-of-distribution (OOD) datasets. To illustrate this, we train one SiCL model on a combined dataset of both SF and ER, and another solely on the SF dataset. The comparative performance of these models is detailed in Tab. A9. The model trained on the combined ER and SF datasets exhibited markedly better performance, not only on the ER dataset but also on the other two OOD datasets, with only a marginal decrease in performance on the SF dataset. These findings suggest that enhancing the diversity of the training data correspondingly improves the model's ability to generalize and maintain robust performance across novel OOD datasets.

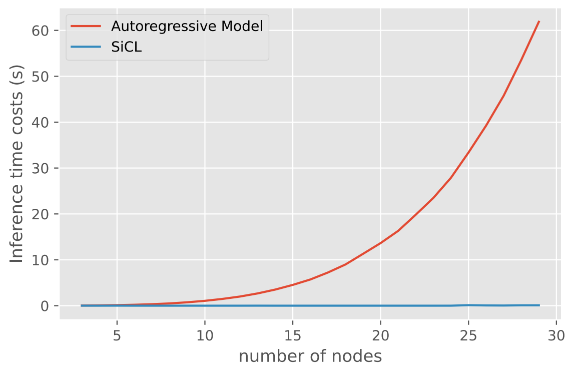
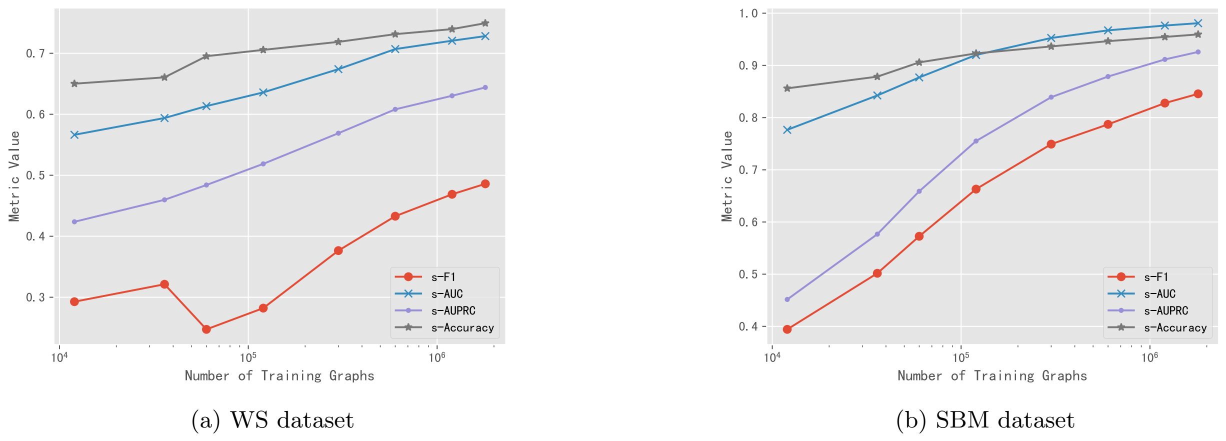


Figure A7: Comparison between an autoregressive model and SiCL on inference time costs.

A8.6 Varying Amount of Training Graphs

We present an analysis of how varying the amount of the training graphs influences performance on the skeleton prediction task. The results, depicted in Fig. A8, illustrate a clear trend: model performance improves in tandem with the expansion of the training dataset. This trend underscores the potential of our method to achieve even greater accuracy given a more extensive dataset.



(a) WS dataset (b) SBM dataset

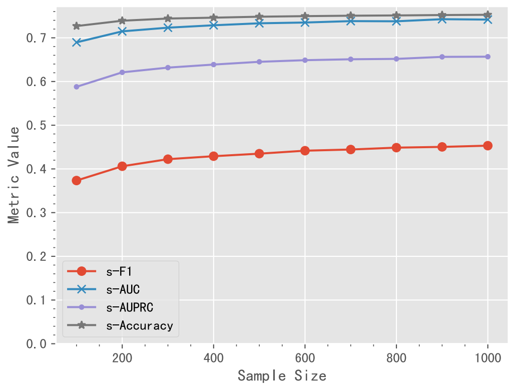
Figure A8: Model performance with varying amount of training graphs.

A8.7 Varying Sample Size

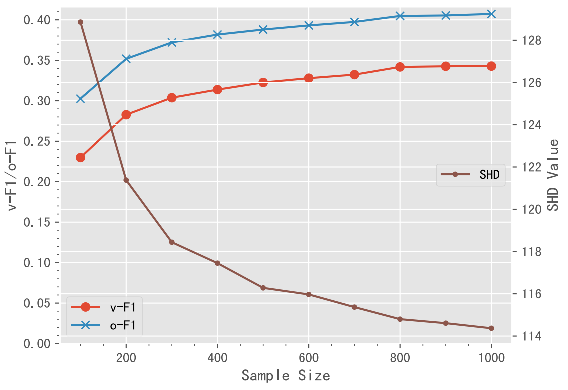
We assess SiCL across various quantities of observational samples per graph during testing (100, 200, ..., 1000). The outcomes for both the skeleton prediction task and the CPDAG prediction task are depicted in Fig. A9. It is evident that the model's performance enhances with the augmentation of sample size. These consistent upward trends suggest that SiCL exhibits stability and is not overly sensitive to changes in sample size.

A8.8 Varying Edge Density

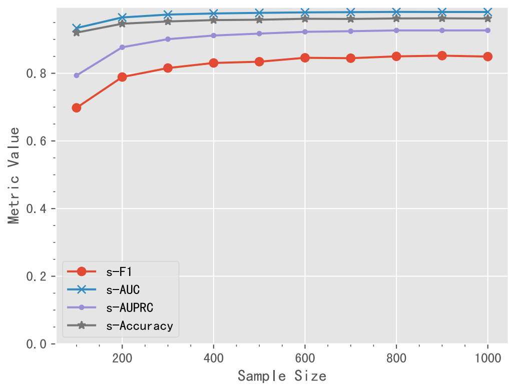
We evaluate SiCL over a range of edge densities in the test graphs, utilizing the SBM dataset, as it allows for the direct setting of average edge densities. The findings are presented in Fig. A10. It's apparent that the task is becomes more difficult as edge densities increase. However, the performance decline is not abrupt, indicating that SiCL's performance remains relatively stable across various edge densities, thereby confirming its versatility.



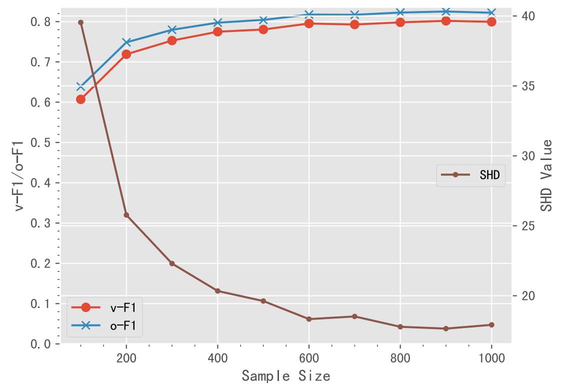
(a) Variation trends of skeleton predicton task performance on WS graph with varying sample sizes.



(b) Variation trends of CPDAG predicton task performance on WS graph with varying sample sizes.



(c) Variation trends of skeleton predicton task performance on SBM graph with varying sample sizes.



(d) Variation trends of CPDAG predicton task performance on SBM graph with varying sample sizes.

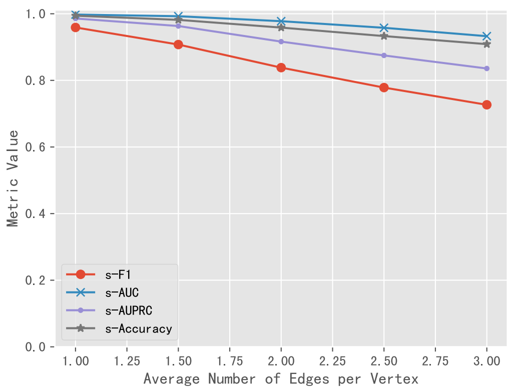
Figure A9: Variation trends of performance with varying sample sizes.

A8.9 Generality on Testing Graph Sizes

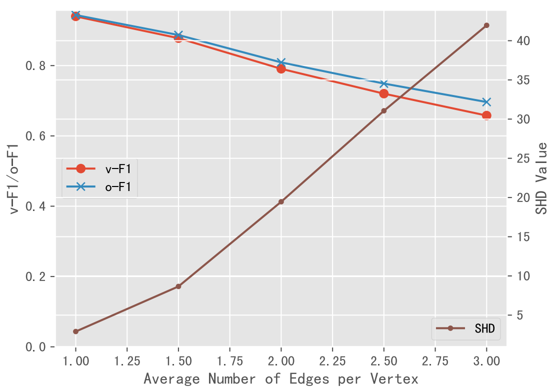
We offer an analytical perspective on the performance of the SiCL model when applied to larger WS-L-G graphs. It is important to highlight that the models were initially trained on graphs comprising 30 vertices, positioning this task within an out-of-distribution setting in terms of graph size. To establish a point of reference, we have included results from the PC algorithm as a baseline comparison. These findings can be examined in Tab. A10. Despite the OOD conditions, SiCL maintains robust performance, reinforcing its scalability and the model's general applicability across varying graph sizes.

A8.10 Acyclicity

We provide an empirical evidence supporting of the rarity of cycles in the predictions. The experimental data presented in Tab. A11 corroborates that cycles are infrequently observed in the predicted CPDAGs, even though without any post-processing on removing cycles.



(a) Variation trends of skeleton predicton task performance on SBM graph with varying edge densities.



(b) Variation trends of CPDAG predicton task performance on SBM graph with varying edge densities.

Figure A10: Variation trends of performance with varying edge densities.

Table A5: **General comparison of SiCL and other methods**. The average performance results in three runs are provided for SiCL method. GES takes more than 24 hours per graph on WS-L-G, and SLdicso is unsuitable on non-linear-Gaussian data, hence the results are not included.

Dataset	Method	Skeleton Prediction				CPDAG Prediction		
		s-F1↑	s-Acc.↑	s-AUC↑	s-AUPRC↑	v-F1↑	o-F1↑	SHD↓
WS-L-G	PC	30.4	65.6	N/A	N/A	15.6	16.0	170.4
	NOTEARS	33.3	65.1	N/A	N/A	27.9	31.5	159.8
	DAG-GNN	35.5	55.4	N/A	N/A	32.2	32.7	193.7
	GRAN-DAG	16.6	62.1	N/A	N/A	11.7	11.7	170.1
	GOLEM	30.0	63.4	N/A	N/A	15.8	19.3	172.7
	SLdisco	0.1	66.0	50.2	34.6	0.0	0.1	147.9
	AVICI	39.9	74.0	71.5	62.2	28.2	35.8	119.2
	SiCL	**44.7**	**75.3**	**73.7**	**65.4**	**32.0**	**38.5**	**116.1**
SBM-L-G	PC	58.8	90.0	N/A	N/A	34.8	35.9	56.4
	GES	70.8	89.4	N/A	N/A	53.9	55.0	60.3
	NOTEARS	80.1	94.5	N/A	N/A	76.2	77.8	26.7
	DAG-GNN	66.2	87.4	N/A	N/A	60.3	62.5	61.0
	GRAN-GAG	22.6	85.9	N/A	N/A	13.8	14.4	64.7
	GOLEM	68.5	88.5	N/A	N/A	63.5	65.2	55.1
	SLdisco	1.9	85.7	56.3	17.6	0.9	1.2	62.6
	AVICI	84.3	96.2	98.1	92.7	79.1	81.6	17.7
	SiCL	**85.8**	**96.4**	**98.3**	**93.4**	**80.6**	**82.7**	**17.1**
WS-RFF-G	PC	36.1	69.9	N/A	N/A	14.8	16.1	156.9
	GES	41.7	66.6	N/A	N/A	21.1	23.6	174.1
	NOTEARS	37.7	64.6	N/A	N/A	30.9	33.4	164.4
	DAG-GNN	33.2	65.4	N/A	N/A	27.0	28.9	161.1
	GRAN-DAG	4.7	66.7	N/A	N/A	0.8	1.1	146.9
	GOLEM	27.6	62.4	N/A	N/A	13.8	17.7	175.8
	AVICI	47.7	75.9	76.3	67.6	38.7	45.2	110.6
	SiCL	**51.8**	**77.4**	**81.1**	**72.9**	**40.3**	**46.3**	**107.0**
SBM-RFF-G	PC	57.5	89.3	N/A	N/A	32.7	34.2	60.9
	GES	56.5	84.9	N/A	N/A	37.0	38.0	82.4
	NOTEARS	55.6	86.2	N/A	N/A	46.5	48.5	66.3
	DAG-GNN	47.1	82.1	N/A	N/A	39.0	40.6	86.2
	GRAN-DAG	17.4	87.4	N/A	N/A	3.2	3.8	58.2
	GOLEM	31.1	75.7	N/A	N/A	23.0	24.8	112.0
	AVICI	76.6	94.5	95.4	85.7	69.3	72.7	27.2
	SiCL	**82.1**	**95.7**	**97.1**	**90.7**	**75.7**	**78.0**	**21.9**
ER-CPT-MC	PC	82.2	83.0	N/A	N/A	39.2	40.6	16.4
	GES	82.1	81.8	N/A	N/A	40.4	42.4	17.1
	NOTEARS	16.7	74.8	N/A	N/A	0.2	0.6	16.1
	DAG-GNN	24.8	73.5	N/A	N/A	3.4	3.7	15.9
	GRAN-DAG	40.8	77.0	N/A	N/A	6.8	7.3	15.6
	GOLEM	37.6	66.4	N/A	N/A	4.6	9.3	21.9
	AVICI	76.9	88.4	93.5	87.9	56.6	57.6	10.2
	SiCL	**84.2**	**90.1**	**96.6**	**94.0**	**58.3**	**59.9**	**10.1**

Table A6: Full ablation study results.

Dataset	Method	s-F1↑	s-Acc.↑	s-AUC↑	s-AUPRC↑	v-F1↑	o-F1↑	SHD↓
WS-L-G	SiCL-Node-Edge	39.9	74.0	71.5	62.2	28.2	35.8	119.2
	SiCL-no-PF	42.4	74.4	72.8	63.5	30.5	37.9	118.4
	SiCL	**44.7**	**75.3**	**73.7**	**65.4**	**32.0**	**38.5**	**116.1**
SBM-L-G	SiCL-Node-Edge	84.3	96.2	98.1	92.7	79.1	81.6	17.7
	SiCL-No-PF	85.5	**96.4**	**98.3**	93.3	79.4	82.2	17.3
	SiCL	**85.8**	**96.4**	**98.3**	**93.4**	**80.6**	**82.7**	**17.1**

Table A7: More performance comparison on the effectiveness of pairwise representation.

Training Dataset	Test Dataset	Method	s-F1↑	s-AUC↑	s-AUPRC↑	s-Acc.↑
ER-L-G	ER-L-G	SiCL-no-PF	75.7	84.6	83.1	78.5
		SiCL	80.2	89.6	90.1	82.3
	SF-L-G	SiCL-no-PF	74.9	92.5	87.3	84.1
		SiCL	79.0	96.0	93.7	87.0
	ER-RFF-G	SiCL-no-PF	49.5	60.5	49.1	58.8
		SiCL	51.0	67.0	57.6	65.2
	SF-RFF-G	SiCL-no-PF	40.4	57.9	38.9	57.5
		SiCL	46.4	71.4	53.7	69.0
SF-L-G	ER-L-G	SiCL-no-PF	64.6	77.3	68.7	70.7
		SiCL	68.0	82.1	76.4	74.3
	SF-L-G	SiCL-no-PF	88.5	96.7	95.0	91.2
		SiCL	89.7	97.9	97.0	92.4
	ER-RFF-G	SiCL-no-PF	44.3	62.3	50.9	58.4
		SiCL	47.0	66.2	55.8	63.3
	SF-RFF-G	SiCL-no-PF	48.1	71.6	53.9	65.8
		SiCL	56.0	79.6	64.8	74.2
ER-RFF-G	ER-L-G	SiCL-no-PF	64.0	73.3	65.8	67.3
		SiCL	72.0	82.0	81.1	75.2
	SF-L-G	SiCL-no-PF	58.1	79.0	66.8	72.8
		SiCL	70.1	88.0	83.6	80.9
	ER-RFF-G	SiCL-no-PF	63.2	74.3	67.7	71.0
		SiCL	74.8	85.7	84.5	79.7
	SF-RFF-G	SiCL-no-PF	56.3	78.3	65.9	75.0
		SiCL	68.2	87.0	81.5	82.1
SF-RFF-G	ER-L-G	SiCL-no-PF	60.3	71.2	58.7	64.7
		SiCL	65.6	78.0	72.5	70.5
	SF-L-G	SiCL-no-PF	73.6	90.5	82.9	81.0
		SiCL	79.1	94.2	90.0	85.5
	ER-RFF-G	SiCL-no-PF	57.7	71.4	60.7	66.8
		SiCL	67.2	80.5	75.8	73.9
	SF-RFF-G	SiCL-no-PF	74.8	90.2	82.4	83.5
		SiCL	80.4	94.2	90.5	87.3

Table A8: Additional Comparison on DAG Prediction

Method	Dataset	F1 Score↑	AUC↑	AUPRC↑	Acc.↑
AVICI	WS-L-G	38.4	86.3	57.7	85.9
SiCL-Corrected AVICI		35.8	87.2	60.5	86.2
AVICI	SBM-L-G	78.1	95.8	80.5	97.3
SiCL-Corrected AVICI		81.3	98.7	90.8	97.8

Table A9: Comparison of SiCL models with different training data diversity on skeleton prediction.

(a) Model trained on both ER and SF

Test Dataset	s-F1↑	s-AUC↑	s-AUPRC↑	s-Acc.↑
WS-L-G	36.3	70.6	60.6	73.3
SBM-L-G	78.1	96.8	88.1	94.8
ER-L-G	80.7	96.0	89.2	94.7
SF-L-G	84.7	98.5	93.6	95.5

(b) Model trained on SF

Test Dataset	s-F1↑	s-AUC↑	s-AUPRC↑	s-Acc.↑
WS-L-G	40.1	63.0	46.1	63.5
SBM-L-G	64.3	91.7	72.9	90.9
ER-L-G	67.1	90.4	73.9	90.8
SF-L-G	87.8	98.9	95.3	96.1

Table A10: Performance comparison with varying amounts of graph sizes.

Metric	s-F1↑			v-F1↑			o-F1↑		
Size	50	70	100	50	70	100	50	70	100
PC	17.7	14.8	10.6	6.4	5.0	3.7	7.0	5.6	4.0
SiCL	**41.6**	**37.4**	**28.3**	**34.9**	**30.7**	**22.6**	**37.9**	**33.7**	**24.8**

Table A11: Count of cycles in the CPDAG predictions without post-processing of removing cycles.

Dataset	WS-L-G	SBM-L-G
Rate of Graphs with Cycles	$0.66 \pm 0.66\%$	$0.00 \pm 0.00\%$