CauScale: Neural Causal Discovery at Scale

Bo Peng[1,2,3] **Sirui Chen**[1,4] **Jiaguo Tian**[1,3] **Yu Qiao**[1,2] **Chaochao Lu**[1*]
[1]Shanghai Artificial Intelligence Laboratory
[2]Shanghai Innovation Institute [3]Shanghai Jiao Tong University [4]Tongji University
*Correspondence: luchaochao@pjlab.org.cn

Abstract

Causal discovery is essential for advancing data-driven fields, yet existing approaches face significant time- and space-efficiency bottlenecks when scaling to large graphs. To address this challenge, we present CauScale, a neural architecture designed for efficient causal discovery that scales inference to graphs with up to 1000 nodes. CauScale improves time efficiency via a reduction unit that compresses data embeddings and improves space efficiency by adopting tied attention weights to avoid maintaining axis-specific attention maps. To keep high causal discovery accuracy, CauScale adopts a two-stream design: a data stream extracts relational evidence from high-dimensional observations, while a graph stream integrates statistical graph priors and preserves key structural signals. CauScale successfully scales to 500-node graphs during training, where prior work fails due to space limitations. Across varying testing datasets, CauScale achieves 99.6% mAP on in-distribution data and 84.4% on out-of-distribution data, while delivering 4×–13,000× inference speedups.

1 INTRODUCTION

Causal discovery aims at uncovering causal relationships and mechanisms from observational data (Spirtes et al., 2000; Pearl, 2009; Glymour et al., 2019). Existing causal discovery algorithms face major time- and space-efficiency bottlenecks, particularly when scaling to large graphs. Constraint-based algorithms (e.g., PC and FCI (Spirtes et al., 2000)) can become time-prohibitive due to large numbers of conditional-independence tests. Score-based methods such as NOTEARS (Zheng et al., 2018) and RL-BIC (Zhu et al., 2019) require solving a fresh continuous optimization problem for each dataset. To reduce runtime, AVICI (Lorch et al., 2022) amortizes causal discovery by pretraining a supervised model on simulated data and performing zero-shot prediction at test time. However, its attention mechanism scales unfavorably with the number of variables, leading to substantial memory pressure on large graphs.

To overcome these time- and space-efficiency bottlenecks, we propose CauScale, an efficient neural architecture for causal discovery. CauScale adopts a two-stream design with a data stream and a graph stream. For efficiency, we introduce a *reduction unit* that compresses the data embeddings during network processing. Moreover, we adopt tied attention weights (Rao et al., 2021): sharing attention weights across axis avoids maintaining axis-specific attention maps and substantially reduces the memory cost. To improve efficiency without sacrificing discovery quality, we further design a *data-graph block* that (i) injects graph-prior and (ii) mitigates information loss from data reduction by distilling key relational signals from data stream into graph stream before reduction.

Extensive experiments demonstrate the superior accuracy and uperior accuracy of CauScale. It achieves an mAP of 99.6% on in-distribution data and 84.4% on out-of-distribution data. It is the fastest method, outperforming previous approaches by 4× to 13,000×. Furthermore, during training, CauScale scales successfully to 500-node graphs, a setting where AVICI fails due to limited memory and excessive space costs.

2 CAUSCALE

2.1 Overall Architecture

Suppose the ground truth causal graph is $G = (V, E)$, let n denote the number of graph nodes and m the number of observational samples. The input data $\mathcal{D} \in \mathbb{R}^{m\times n\times 2}$ concatenates observational variables $D \in \mathbb{R}^{m\times n}$ and a binary intervention indicator $I \in \{0,1\}^{m\times n}$, where $I = 1$ indicates that variable is intervened. The model takes $\mathcal{D}$ and a statistical graph

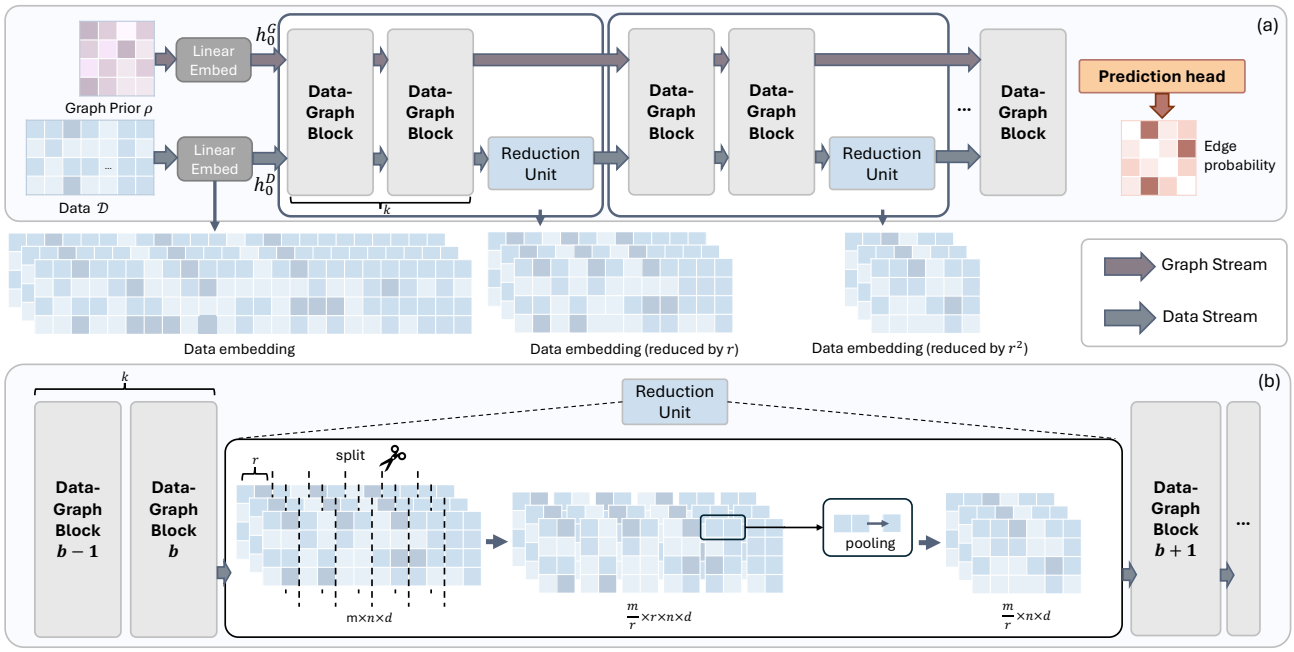

Figure 1: The architecture of CauScale. (a) The overall architecture and the changes of data embedding size during network processing. (b) The reduce operation in *reduction unit*. Between each k *data-graph block*s, the *reduction unit* pool the data embedding along the observation dimension to reduce it with a fraction of r.

prior $\rho \in \mathbb{R}^{n\times n}$ computed from D as inputs, and outputs a probabilistic adjacency matrix $\hat{G} \in \mathbb{R}^{n\times n}$ representing the likelihood of directed causal relations. The prior ρ is defined as the inverse covariance matrix.

Shown in Figure 1, the inputs $\mathcal{D}$ and ρ are encoded into initial embeddings $h^{\mathcal{D}} \in \mathbb{R}^{m\times n\times d}$ and $h^{G} \in \mathbb{R}^{n\times n\times d}$ via linear layers, where d is the embedding dimension. These embeddings are then processed by alternating stacks of *data-graph block* and *reduction unit*. Each *data-graph block* updates both the data and graph streams (Section 2.2). Every k blocks, the *reduction unit* pools the data-stream embedding along the sample dimension to reduce its length by a factor of r (Section 2.3). Within each *data-graph block*, we employ tied attention weights to reduce memory overhead (Section 2.4). Finally, the graph-stream output is normalized and fed into a prediction head adopted from Lippe et al. (2021) to produce $\hat{G}$.

2.2 DataGraph Block

As shown in Figure 2, each *data-graph block* consists of three modules: a data layer, a data2graph layer, and a graph layer. Given the incoming data and graph embeddings (h^D_{b-1}, h^G_{b-1}) before block $b \in \{0, \ldots, B-1\}$, the block proceeds in three steps: (1) The data layer updates the data stream embedding, producing h^D_b. This updated embedding is forwarded to the next *data-graph block* or to the *reduction unit* for compression. (2) The data2graph layer extracts pairwise relational evidence from the data stream and summarizes it into a graph message. Given the data embedding $h^D_b \in \mathbb{R}^{m\times n\times d}$, we first apply a data axial-attention layer to obtain $h^{D\to G} \in \mathbb{R}^{m\times n\times d}$. We then map $h^{D\to G}$ to two node-level embeddings $u^{D\to G}, v^{D\to G} \in \mathbb{R}^{n\times d}$ using two separate PoolingFFN modules. Finally, we form: $\omega^{D\to G}_b = u^{D\to G}\,(v^{D\to G})^\top \in \mathbb{R}^{n\times n}$, which represents directed pairwise relations between variables. (3) The graph layer injects $\omega^{D\to G}_b$ into the graph stream by concatenating it with the previous graph embedding $h^G_{b-1} \in \mathbb{R}^{n\times n\times d}$, yielding $h^{G'}_b \in \mathbb{R}^{n\times n\times (d+1)}$. A linear projection maps $h^{G'}_b$ back to $\mathbb{R}^{n\times n\times d}$, which is then processed by a graph axial-attention layer to produce the updated graph embedding $h^G_b \in \mathbb{R}^{n\times n\times d}$.

2.3 Reduction Unit

CauScale applies the *reduction unit* every k *data-graph block*s. Given a data embedding h^D_b after block b and a reduction factor r, we group the observation dimension into chunks of size r and average-pool within each chunk. When $r \nmid m$, we set $\hat{m} = r\lfloor m/r \rfloor$ and discard the last $m - \hat{m}$ samples for convenience (replacing m with $\hat{m}$). Specifically, we reshape $h^D_b : m \times n \times d \to \frac{m}{r} \times r \times n \times d$, and apply average pooling over the group dimension of size r, yielding the reduced embedding $\tilde{h}^D_b \in \mathbb{R}^{\frac{m}{r}\times n\times d}$. With a reduction factor r applied every k blocks, the effective sample length at

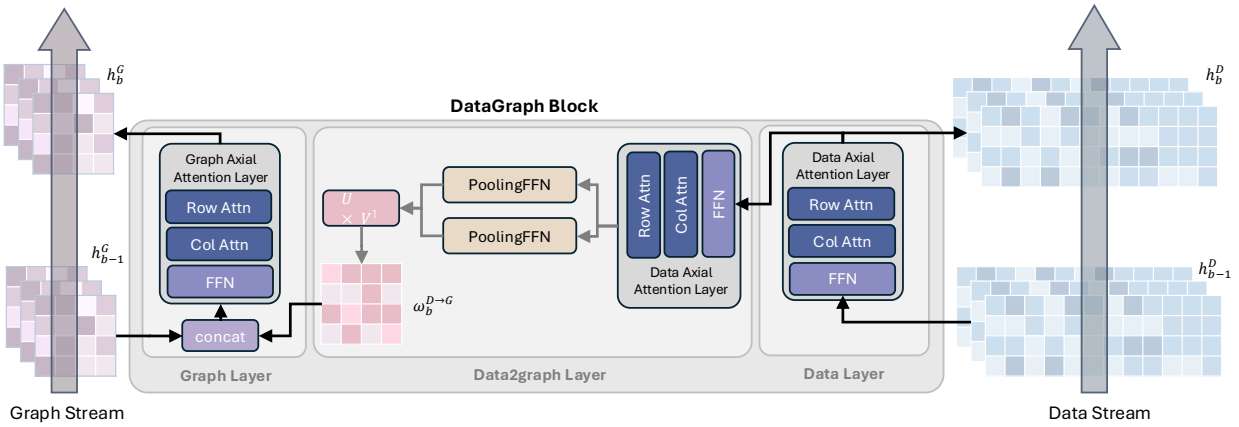

Figure 2: Structure of the *DaraGraph Block*. The *data-graph block* process information on data and graph stream. On data stream, after being processed by the data axial attention layer, data embedding h_b^D is sent to both the next module on data stream and summarized by the data2graph layer to graph message $\omega_b^{D\to G}$. The message will be concatenated with previous graph embedding h_{b-1}^G and processed by graph layer in graph stream.

block b becomes $m_b = m/r^{\lfloor b/k \rfloor}$. As the dominant cost of data-stream axial attention comes from *sample-axis* attention with cost $\mathcal{O}(nm^2)$ and *node-axis* attention with cost $\mathcal{O}(mn^2)$, the average per-block compute is therefore

$$\mathcal{C}_{\text{sample}} \propto \frac{1}{B}\sum_{b=0}^{B-1} nm_b^2 = \frac{nm^2}{B}\sum_{b=0}^{B-1} r^{-2\lfloor b/k \rfloor},$$
$$\mathcal{C}_{\text{node}} \propto \frac{1}{B}\sum_{b=0}^{B-1} n^2 m_b = \frac{n^2 m}{B}\sum_{b=0}^{B-1} r^{-\lfloor b/k \rfloor}.$$

In our experiments (B=10, k=2, r=2), this yields only 26.64% of the baseline sample-axis compute and 38.75% of the baseline node-axis compute.

2.4 Tied Attention Weights

The core component in each layer of a *data-graph block* is an axial-attention layer, which applies self-attention along two axis (row and column), followed by an FFN. To improve space efficiency, we adopt the tied attention weight mechanism from Rao et al. (2021), which avoids maintaining axis-specific attention maps and substantially reduces attention memory. Given attention on row axis, standard attention mechanism stores axis-specific attention maps $A \in \mathbb{R}^{R\times H\times C\times C}$, resulting in $\mathcal{O}(RHC^2)$ memory. With tied attention weights (Rao et al., 2021), attention weights are shared across target axis and only $A \in \mathbb{R}^{H\times C\times C}$ is stored, reducing attention-map memory to $\mathcal{O}(HC^2)$. Analogously, for column-axis attention, the memory cost is reduced from $\mathcal{O}(CHR^2)$ to $\mathcal{O}(HR^2)$.

3 EXPERIMENT

3.1 Baselines and Metrics

We evaluate our approach against several baselines spanning different paradigms: (1) constraint-based methods: Fast Causal Inference (FCI) (Spirtes et al., 2013); (2) score-based methods: NOTEARS (Zheng et al., 2018), SDCD (Nazaret et al., 2023) (3) FCM-based methods: DiffAN (Sanchez et al., 2023) (4) pre-training- based methods: AVICI (Lorch et al., 2022) and SEA (Wu et al., 2025). We adopt four standard metrics for causal structure learning: Structural Hamming Distance (SHD), Mean Average Precision (mAP), Area Under the ROC Curve (AUC), and Orientation Accuracy (OA). To evaluate causal discovery efficiency, we report inference time.

3.2 Datasets

We consider two types of data: synthetic datasets generated from SCMs and semi-synthetic single-cell expression datasets simulated with SERGIO GRN (Dibaeinia and Sinha, 2020).

Training Data. For synthetic data, we generate datasets based on Erdős-Rényi and Scale-Free graphs. The graph size n ranges from 10 to 500, with edge counts $|E| \in \{n, 2n, 3n, 4n\}$. We include both linear and neural network (NN) functions with additive or non-additive Gaussian noise. For each graph, we sample 1,000 observations, with observational and interventional data in a $1:n$ ratio. For single-cell GRNs, the graphs are initialized using Erdős-Rényi, Scale-Free, and Stochastic Block Models. Given the complexity of gene regulatory dynamics, we increase the sample size to 5,000 to ensure reliable structure learn-

Model	Linear				NN non-add.				Sigmoid†				Polynomial†				Time
	mAP	SHD	AUC	OA	mAP	SHD	AUC	OA	mAP	SHD	AUC	OA	mAP	SHD	AUC	OA	(s)
Synthetic (sample size= 1000)																	
Setting: $n = 100, \lvert E\rvert = 400$																	
FCI	12.4	372.0	55.3	10.7	11.1	359.8	55.3	11.0	15.1	348.2	56.3	12.6	9.0	368.4	53.0	6.1	84.987
NOTEARS	29.4	300.8	51.8	27.8	17.8	337.0	50.4	19.5	11.3	366.0	49.1	8.3	8.6	371.8	52.5	4.9	2170.2
SDCD	42.3	400.2	89.3	81.6	65.7	272.6	89.0	79.8	61.9	327.8	87.6	76.5	41.9	303.8	70.9	42.8	67.428
DiffAN	10.6	475.4	51.3	8.5	8.4	465.2	53.6	9.3	12.3	389.4	57.4	12.3	11.6	378.9	50.3	11.1	1973.4
AVICI	25.9	394.0	80.2	81.4	32.3	361.2	81.1	81.5	22.7	371.8	68.4	63.2	5.6	384.6	45.9	36.8	0.2974
SEA-gies	92.1	108.6	99.2	94.8	51.2	306.2	88.4	86.6	72.7	192.2	92.2	85.4	36.2	319.2	74.2	70.8	8.7759
CauScale	**99.6**	**15.2**	**100.0**	**100.0**	**89.0**	**105.6**	**98.5**	**99.5**	**84.4**	**125.8**	**95.0**	**94.6**	**50.3**	**252.2**	**79.4**	**81.7**	**0.0384**
Setting: $n = 1000, \lvert E\rvert = 2000$†																	
FCI	32.9	1309.0	67.2	34.4	12.7	1721.2	58.7	17.4	8.6	1828.6	54.5	9.0	1.5	2008.8	50.7	1.4	2005.2
NOTEARS	30.5	1388.6	50.6	30.5	20.5	1677.8	50.0	23.2	11.0	1790.4	50.0	10.9	7.3	1893.2	53.6	7.1	10896
SDCD	54.1	1793.2	99.3	98.6	59.6	2015.0	87.9	76.0	48.5	1649.2	89.2	78.5	**29.8**	**1798.4**	74.6	49.2	65.386
AVICI	0.2	1985.8	47.5	46.2	0.9	1980.2	56.8	54.6	0.2	2006.8	39.6	41.9	0.1	2037.0	37.0	40.3	3.3407
SEA-gies	66.3	2944.8	98.2	80.2	11.9	3227.4	73.9	66.2	48.1	1359.0	88.8	70.1	20.6	6814.2	72.0	58.9	218.23
CauScale	**96.6**	**230.0**	**100.0**	**96.5**	**79.7**	**835.0**	**98.2**	**96.6**	**64.5**	**1064.6**	**95.3**	**79.0**	18.9	3985.0	**78.1**	**59.7**	**0.8288**

Model	$n = 100, \lvert E\rvert = 400$					$n = 200, \lvert E\rvert = 400$				
	mAP	SHD	AUC	OA	Time	mAP	SHD	AUC	OA	Time
SERGIO-GRN (sample size= 20000)										
NOTEARS	4.1	492.8	50.5	2.0	5040.8	1.0	591.0	49.8	0.4	5805.9
SDCD	4.0	764.6	49.0	5.1	579.68	2.1	1265.0	49.9	3.7	670.10
AVICI	Out of memory					Out of memory				
SEA-gies	5.6	400.2	58.8	61.6	11.769	1.2	407.8	55.0	57.07	17.288
CauScale	**71.4**	**290.8**	**95.1**	**94.2**	**1.3058**	**34.5**	**336.8**	**90.3**	**90.9**	**2.5195**

Table 1: Model performance comparison. † indicates out-of-distribution settings. Time represents inference time.

ing. Consequently, to balance the computational overhead introduced by this larger sample size, we restrict the maximum graph size to $n = 200$.

Tesing Data. For synthetic data, we assess the model on graphs with $(n, |E|) \in \{(100, 400), (1000, 2000)\}$ and a sample size of 1,000. We introduce two out-of-distribution causal mechanisms: sigmoid and polynomial functions. For GRN data, we use graphs with $(n, |E|) \in \{(100, 400), (200, 400)\}$, and a sample size of 20,000. For each configuration, we generate 5 Erdős-Rényi graphs and report the averaged results.

3.3 Model Performance

As illustrated in Table 1, on the synthetic linear dataset with $n = 100$, our model achieves near-perfect causal discovery on linear data (99.6% mAP). On larger-scale graphs with 1000 nodes (out-of-distribution size), CauScale maintains high performance (96.6% mAP for linear). CauScale also exhibits strong generalization capabilities on out-of-distribution mechanisms. Specifically, on the polynomial dataset ($n = 100$) with more complex mechanisms, CauScale achieves 50.3% mAP, whereas the second and third best methods, SEA and SDCD, drop to 36.2% and 41.9%, respectively. On the GRN dataset, CauScale achieves the best performance across all metrics and settings among all baselines.

Moreover, CauScale achieves the shortest inference time. Even on graphs with $n = 1000$, our inference takes less than 1 second (0.8288s), achieving a speedup of over 13,000× compared to NOTEARS (10,896s), 200× compared to SEA-gies (218s), and 4× compared to AVICI (3.34s). Regarding space efficiency, on SERGIO-GRN dataset, AVICI fails with an Out-of-Memory error even at $n = 100$. In contrast, CauScale successfully scales to $n = 200$ with 20,000 samples.

4 CONCLUSION

We present CauScale, an efficient neural architecture for large-scale causal discovery that addresses the time and memory bottlenecks of prior methods. CauScale combines a reduction unit for time efficiency, tied attention weights for space efficiency, and a two-stream design that preserves structural signals under compression. Extensive experiments on synthetic and semi-synthetic single-cell benchmarks show that CauScale scales training to 500-node graphs and enables inference on graphs with up to 1,000 nodes, achieving strong accuracy with 4×–13,000× speedups over existing approaches.

Acknowledgements

This work is supported in part by the Shanghai Artificial Intelligence Laboratory and in part by the National Natural Science Foundation of China under Grant 625B2131.

References

Dibaeinia, P. and Sinha, S. (2020). Sergio: a single-cell expression simulator guided by gene regulatory networks. *Cell systems*, 11(3):252–271.

Glymour, C., Zhang, K., and Spirtes, P. (2019). Review of causal discovery methods based on graphical models. *Frontiers in genetics*, 10:524.

Lippe, P., Cohen, T., and Gavves, E. (2021). Efficient neural causal discovery without acyclicity constraints. *arXiv preprint arXiv:2107.10483*.

Lorch, L., Sussex, S., Rothfuss, J., Krause, A., and Schölkopf, B. (2022). Amortized inference for causal structure learning. *Advances in Neural Information Processing Systems*, 35.

Nazaret, A., Hong, J., Azizi, E., and Blei, D. (2023). Stable differentiable causal discovery. In *Forty-first International Conference on Machine Learning*.

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

Rao, R., Liu, J., Verkuil, R., Meier, J., Canny, J. F., Abbeel, P., Sercu, T., and Rives, A. (2021). Msa transformer. *bioRxiv*.

Sanchez, P., Liu, X., O'Neil, A. Q., and Tsaftaris, S. A. (2023). Diffusion models for causal discovery via topological ordering. In *The Eleventh International Conference on Learning Representations*.

Spirtes, P., Glymour, C., and Scheines, R. (2000). Causation, prediction, and search. adaptive computation and machine learning series. *The MIT Press*, 49:77–78.

Spirtes, P. L., Meek, C., and Richardson, T. S. (2013). Causal inference in the presence of latent variables and selection bias. *arXiv preprint arXiv:1302.4983*.

Wu, M., Bao, Y., Barzilay, R., and Jaakkola, T. S. (2025). Sample, estimate, aggregate: A recipe for causal discovery foundation models. *Trans. Mach. Learn. Res.*, 2025.

Zheng, X., Aragam, B., Ravikumar, P., and Xing, E. P. (2018). DAGs with NO TEARS: Continuous Optimization for Structure Learning. In *Advances in Neural Information Processing Systems*.

Zhu, S., Ng, I., and Chen, Z. (2019). Causal discovery with reinforcement learning. *arXiv preprint arXiv:1906.04477*.

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/No/Not Applicable] Yes, see Section 2.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes/No/Not Applicable] Yes, see Section 2 and 3.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes/No/Not Applicable]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes/No/Not Applicable] Not Applicable.
   (b) Complete proofs of all theoretical results. [Yes/No/Not Applicable] Not Applicable.
   (c) Clear explanations of any assumptions. [Yes/No/Not Applicable] Not Applicable.

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/No/Not Applicable] No, the code will be released later.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes/No/Not Applicable] Yes, see supplemental materials.
   (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/No/Not Applicable] Yes, see Section 3.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes/No/Not Applicable] Yes, see supplemental materials.

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/No/Not Applicable] Yes, see Section 3 and supplemental materials.
   (b) The license information of the assets, if applicable. [Yes/No/Not Applicable] Not Applicable.

(c) New assets either in the supplemental material or as a URL, if applicable. [Yes/No/Not Applicable] Yes, see supplemental material.

(d) Information about consent from data providers/curators. [Yes/No/Not Applicable] Not Applicable.

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes/No/Not Applicable] 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. [Yes/No/Not Applicable] Not Applicable.

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Yes/No/Not Applicable] Not Applicable.

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Yes/No/Not Applicable] Not Applicable.

Supplementary Materials

1 RELATED WORK

Existing causal structure learning methods can be broadly categorized into *non-amortized* and *amortized (zero-shot)* approaches, relying on whether inference on a new dataset requires solving a dataset-specific optimization problem.

Non-amortized causal discovery. Non-amortized methods perform causal discovery by solving an optimization or search problem independently for each dataset, leading to high computational cost and limited scalability. 1) **Constraint-based algorithms**, such as PC and FCI (Spirtes et al., 2000), infer graph structures via conditional independence tests but suffer from exponential complexity as the number of variables grows. 2) **Score-based algorithms** optimize a predefined score over the space of graph structures (Tsamardinos et al., 2006; Goudet et al., 2017). Classical approaches rely on greedy combinatorial search, including GES (Chickering, 2002) and GIES (Hauser and Bühlmann, 2012). To improve scalability, recent work reformulates causal discovery as continuous optimization with differentiable acyclicity constraints (Zheng et al., 2018; Lachapelle et al., 2019; Ke et al., 2019; Zhu et al., 2019; Brouillard et al., 2020). NOTEARS (Zheng et al., 2018) introduces a smooth acyclicity constraint, while SDCD (Nazaret et al., 2023) further improves stability via spectral constraints and staged optimization. 3) **Functional Causal Model (FCM) based methods** exploit asymmetries in the data-generating process for identifiability. Early approaches such as LiNGAM (Shimizu et al., 2006) rely on Independent Component Analysis (ICA) (Hyvärinen et al., 2001), whereas recent methods integrate deep generative models. For example, DiffAN (Sanchez et al., 2023) frames causal discovery as topological sorting using diffusion-based score estimators. Despite their diversity, non-amortized methods require dataset-specific optimization, making them computationally expensive and unsuitable for large-scale or real-time inference.

Amortized (zero-shot) causal discovery. Amortized approaches aim to eliminate dataset-specific optimization by learning a shared inference model that maps datasets directly to causal graphs, enabling zero-shot inference on unseen data (Lorch et al., 2022; Ke et al., 2023; Wu et al., 2025; Dhir et al., 2025). Lorch et al. (2022); Ke et al. (2023) pioneer amortized variational inference for causal discovery. However, these methods rely on high-dimensional embeddings that scale poorly with graph size. SEA (Wu et al., 2025) mitigates this issue by decomposing large graphs into subproblems, but its reliance on classical estimators such as GIES and extensive sub-batch sampling limits inference speed and causes information loss across variable partitions.

2 DETAILS OF THE TIED ATTENTION MECHANISM

For illustration, consider attention along row-axis with $Q, K, V \in \mathbb{R}^{R\times C\times H\times d_{\text{head}}}$, where R and C denote the row and column dimensions (e.g., $R{=}m$, $C{=}n$ for the data stream), H is the number of heads, and d_{head} is the head dimension. Following Rao et al. (2021), we tie attention weights across rows and only store $A \in \mathbb{R}^{H\times C\times C}$, while keeping the output shape unchanged:

$$A_{h,i,j} = \sum_{r=1}^{R}\sum_{t=1}^{d_{\text{head}}} Q_{r,i,h,t}\cdot K_{r,j,h,t},$$

$$O_{r,i} = W^{O}\cdot\left[\sum_{j=1}^{C}\text{softmax}_j(A_{h,i,j})\cdot V_{r,j,h,:}\right]_h + b^{O},$$

3 DETAILS OF THE PREDICTION HEAD

After the final *data-graph block*, we take the graph-stream output $h^G_{B-1} \in \mathbb{R}^{n\times n\times d}$, apply layer normalization, and feed it into a pairwise graph prediction head. Following Wu et al. (2025) and Lippe et al. (2021), we do not explicitly enforce acyclicity during prediction, since imposing DAG constraints typically requires additional constrained optimization or post-processing and can be computationally expensive. Moreover, real-world data sometimes contain cycles. We adopt the decomposed head in Lippe et al. (2021). For each unordered node pair $\{i, j\}$ with $i < j$, we compute logits over three edge states (no edge, $i \to j$, $j \to i$) by

$$g_{\{i,j\}} = \mathrm{FFN}\big([h^G_{B-1,i,j},\, h^G_{B-1,j,i}]\big) \in \mathbb{R}^3, \tag{1}$$

where $[\cdot,\cdot]$ denotes concatenation. Collecting all pairs yields $g \in \mathbb{R}^{\frac{N(N-1)}{2}\times 3}$, and we obtain probabilities via a softmax over the three states for each pair. In our experiments, this decomposed head achieves accuracy comparable to the AVICI prediction head while empirically producing fewer cycles in the decoded graphs.

4 EVALUATION METRICS

To evaluate causal discovery performance, we adopt four standard metrics for causal structure learning: (1) **Structural Hamming Distance (SHD)**: the minimum number of edge insertions, deletions, and reversals required to transform the predicted graph into the ground-truth graph. (2) **Mean Average Precision (mAP)**: the area under the precision-recall curve computed over all candidate edges, averaged across the graph. (3) **Area Under the ROC Curve (AUC)**: the area under the ROC curve computed over all candidate edges, averaged across the graph. (4) **Orientation Accuracy (OA)**: the fraction of ground-truth directed edges for which the model assigns a higher probability to the correct direction.

5 DATA GENERATION DETAILS

5.1 Causal Graph Details

We evaluate our method on various causal graphs. Below, we provide a detailed description of the graph models used in this study. Both the synthetic data and SERGIO-GRN data are generated based on these Directed Acyclic Graph (DAG) structures.

- **Erdős-Rényi (ER):** A standard random graph model where edges are added between any pair of nodes with a fixed probability p. This results in a graph where the degree distribution is approximately Poissonian, representing networks with uniform connectivity patterns.
- **Scale-Free (SF):** Generated using the Barabási-Albert preferential attachment process. New nodes are more likely to attach to existing nodes with high degrees. This topology creates networks with "hubs" and follows a power-law degree distribution, simulating real-world biological (e.g., gene regulatory networks) or social networks.
- **Stochastic Block Model (SBM):** A generative model for graphs with community structure. Nodes are assigned to one of K latent blocks (clusters). Edge probabilities depend on the block membership of the nodes (high probability within blocks, low probability between blocks). This is particularly useful for modeling modular systems, such as protein-protein interaction networks with functional modules.

5.2 Synthetic Data Generation

The details of the distribution settings for the synthetic data are explained below. We generate data following the code and settings in Wu et al. (2025) and Brouillard et al. (2020). Let X_i denote the target node, PA_i its parents, N_i an independent noise variable, and W the randomly initialized weights.

- **Linear**: The most fundamental assumption where dependencies are linear: $X_i = W_i\mathrm{PA}_i + N_i$.
- **Neural Networks (NN-Add)**: The mechanism follows $X_i = \mathrm{MLP}(\mathrm{PA}_i) + N_i$, where MLP is a random initialized Multi-Layer Perceptron (MLP) with a single hidden layer and nonlinear activations (PReLU).

- **Neural Networks (NN)**: The noise is concatenated with the parents as input to the neural network: $X_i = \text{MLP}(\text{PA}_i, N_i)$.

- **Sigmoid Additive**: $X_i = \sum W_j \sigma(PA_{ij}) + N_i$, simulating biological saturation effects.

- **Polynomial**: $X_i = \sum_{k=0}^{2} W_k \text{PA}_i^k + N_i$, modeling polynomial dependencies.

Root nodes are initialized using a Uniform distribution of Uniform(-1,1). We set noise to $N_i \sim 0.4 \cdot \mathcal{N}(0, \sigma^2)$, where $\sigma^2 \sim \text{Uniform}(1, 2)$. We apply interventions (hard intervention) one node at a time, covering all node and setting their mechanisms to Uniform(-1,1). For smaller graphs ($N \in \{10, 20, 100\}$), we generate 600 distinct graph structures for each parameter combination. For larger graphs ($N \in \{150, 200, 300, 500\}$), we generate 300 distinct structures per combination due to the increasing computational time required for generation.

5.3 SERGIO-GRN Data Generation

We generated the data using a slightly modified version of the SERGIO-GRN (Dibaeinia and Sinha, 2020) code from AVICI (Lorch et al., 2022). The simulator generates gene expression data by sampling from the steady state of a dynamic system, described by Stochastic Differential Equations (SDEs) (Dibaeinia and Sinha, 2020). Downstream regulatory interactions are modeled using Hill functions (Chu et al., 2009), ensuring the realistic gene behavior. All samples are generated under gene intervention settings. Specifically, we conduct gene knockouts by setting the target gene expression level to zero. Regarding graph structures, we employ Erdős-R'enyi (ER), Scale-Free, and Stochastic Block Models for training. The number of cell types is set between 5 and 10. We generate 200 distinct graph structures for each setting. We consider training graphs with sizes $N \in \{10, 20, 30, 50, 80, 100, 150, 200\}$, where the number of edges $E \in \{2N, 4N, 6N\}$.

6 BASELINE IMPLEMENTATION DETAILS

INVCOV and CORR For both INVCOV and CORR, we discretize the predicted continuous values to match the sparsity of the ground truth. The threshold is set to the $(1 - \frac{e}{n^2})$-th quantile of the predictions, where e and n represent the number of edges and nodes in the ground truth, respectively.

FCI We implement the Fast Causal Inference (FCI) algorithm using the *causal-learn* library[1]. FCI is a constraint-based causal discovery algorithm that identify causal relationships in the presence of latent confounders and selection bias. We use Fisher-Z test with $\alpha = 0.05$ significance Level during experiment.

NOTEARS We utilize the official implementation of the NOTEARS algorithm[2]. Following the default setting in the repository, we apply a threshold of 0.3 to the estimated weight matrix to filter out weak edges before computing the Structural Hamming Distance (SHD).

DiffAN We implemented DiffAN by adopting the official hyperparameter configurations, which the original authors (Sanchez et al., 2023) noted are largely hard-coded and robust across diverse datasets. To strictly adhere to the non-approximated version of the algorithm, the `residue` parameter was set to True. For downstream evaluation requiring continuous scores such as AUC and mAP, we extracted the edge existence p-values and applied a $-\log_{10}$ transformation to derive the final confidence estimates. For metrics requiring a binary adjacency matrix such as SHD, we followed the hyperparameters $\alpha = 0.05$ as the threshold for edge pruning.

SDCD We utilized the official implementation of SDCD[3]. All hyperparameters followed the default settings provided in the official repository. We specified GPU as the computing device to ensure efficiency. To compute SHD, we apply a discretization threshold of 0.5.

[1] `https://causal-learn.readthedocs.io`
[2] `https://github.com/xunzheng/notears`
[3] `https://github.com/azizilab/sdcd`

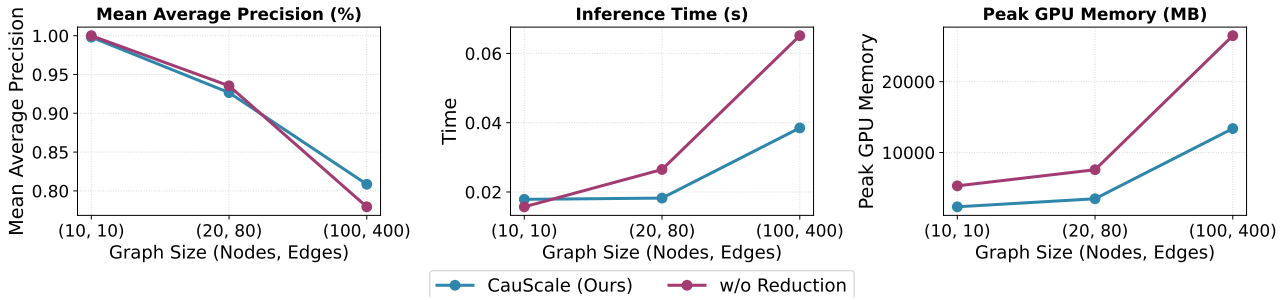

Figure 1: Comparison of w/ and w/o Reduction Unit.

AVICI Due to the high memory requirements encountered when attempting to train AVICI on our datasets (resulting in Out-of-Memory errors), we utilized the pre-trained checkpoints provided in the official repository[4]. For synthetic data, we employed the `scm-v0` model, which was pre-trained on diverse linear and non-linear datasets. For the SERGIO-GRN dataset, we utilized the `neurips-grn` checkpoint. All other settings remained consistent with our experimental setup, utilizing both interventional and observational data. We apply a discretization threshold of 0.5.

SEA We trained SEA using the same training data as ours, including both synthetic and SERGIO-GRN datasets. We use the GIES-based architecture. All training and testing configurations followed the default settings of SEA. We apply a discretization threshold of 0.5.

7 CAUSCALE IMPLEMENTATION DETAILS

Model Configuration. The model consists of 10 layers with 128-dimensional embeddings and 16 attention heads. This configuration was determined through hyperparameter tuning across layers $\in \{8, 10\}$ and embedding dimensions $\in \{64, 128, 256\}$.

Data Preprocessing. Each set of data $\mathcal{D}$ is standardized variable-wise. For each variable x_i, we compute the normalized value via $\hat{x}_i = (x_i - \mu_i)/\sigma_i$, where μ_i represents the empirical mean and σ_i is the standard deviation.

Hardware Details All training and inference tasks are conducted on NVIDIA H200 GPUs (141GB memory per GPU) and 164 CPU cores. Note that the baselines (e.g., AVICI) detailed in Section 6 are also evaluated in this environment.

Training Strategy We use the Adam optimizer with a learning rate of 1×10^{-4}. Training is performed on 8 GPUs using distributed data parallelism. For synthetic data, we adopt a two-stage training strategy. This design is motivated by two key factors: (1) It allows the model to capture fundamental causal relationships on simpler graphs (10–100 nodes) before generalizing to complex structures (up to 500 nodes). (2) Grouping graphs by size significantly reduces memory waste caused by excessive zero-padding when batching graphs of vastly different scales (e.g., mixing 10-node and 500-node graphs). Based on this, the training proceeds as follows:

- Stage 1 (10–100 nodes): Batch size of 8 for 37 hours.
- Stage 2 (150–500 nodes): Batch size of 1 for 2.75 hours.

For the SERGIO-GRN dataset, as the graph sizes vary within a narrower range (10–200 nodes), we train the model in a single stage with a batch size of 1 for 44 hours.

[4] `https://github.com/larslorch/avici`

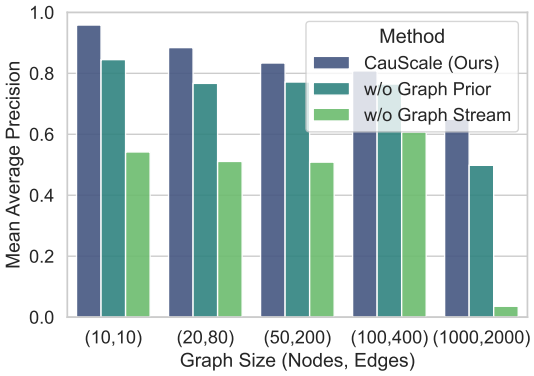

Figure 2: Advantage of *data-graph block* over the block containing the data layer only.

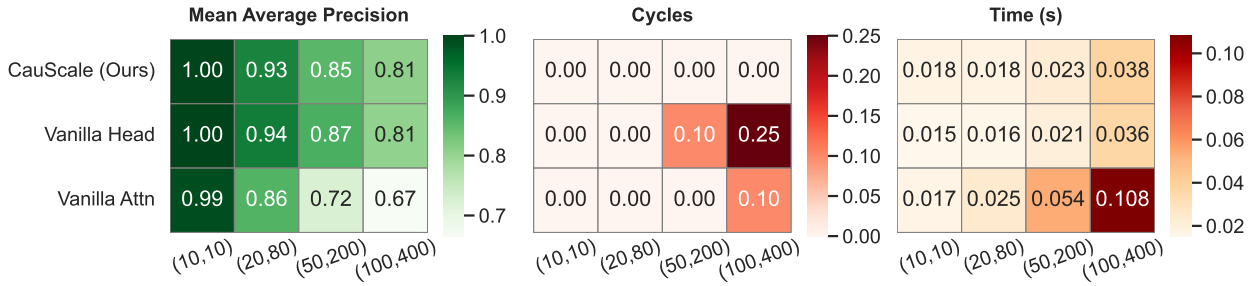

Figure 3: Ablation on components: Ours vs. AVICI.

8 ADDITIONAL EXPERIMENTS

8.1 Ablation Studies

W/ and w/o reduction unit We remove the *reduction unit*and retrain the model on synthetic dataset to validate its importance. Since the network encounters Out-of-Memory errors on the original training set without the *reduction unit*, we use a training subset with node number limited to $n \in \{10, 20, 100\}$. We train both `CauScale` and the architecture w/o *reduction unit*on this subset and evaluate their performance on synthetic test set with the same node number. Other settings in test set are the same with the evaluation benchmark. Results are averaged across all four distributions (linear, NN, sigmoid, and polynomial). Figure 1 illustrates the mean average precision, inference time, and peak GPU memory usage. The benefits of the *reduction unit* become increasingly pronounced as the number of nodes increases, enabling the model to maintain high accuracy while achieving significantly faster inference speeds and lower GPU memory usage.

Graph components We conduct ablation studies by (1) removing the input graph prior by setting the graph input to an all-ones vector (w/o Graph Prior) and (2) removing the graph stream while retaining only the data stream (w/o Graph Stream). We retrain the two ablation versions on our synthetic train set. Figure 2 demonstrates the performance comparison on our synthetic benchmark. Removing the graph stream causes the most significant performance degradation, highlighting the importance of it. Removing the graph prior causes less performance degradation but still yields inferior results compared to our full model, validating the importance of the inductive bias.

Attention shape and output head We conduct an ablation study to analyze the components of our model relative to AVICI. Specifically, we evaluate two variants: (1) replacing the tied-attention mechanism in `CauScale` with the vanilla attention from (Lorch et al., 2022) (Vanilla Attn), and (2) replacing our prediction head in Equation 1 with the vanilla prediction head from (Lorch et al., 2022) (Vanilla Head). The latter projects the graph embedding $h^G_{B-1} \in \mathbb{R}^{n \times n \times d}$ to a logit $g \in \mathbb{R}^{n \times n}$ using a Feed-Forward Network, followed by a sigmoid function to obtain edge probabilities. Due to the high space complexity of vanilla self-attention, we limited

(Nodes, Edges)	Max	Strided	Average
(10,10)	94.3	100.0	100.0
(20,80)	76.0	84.6	92.6
(50,200)	71.2	78.5	85.4

Table 1: Mean Average Precision (%) comparison across different pooling strategies in the Reduction Unit. We use average pooling in `CauScale`.

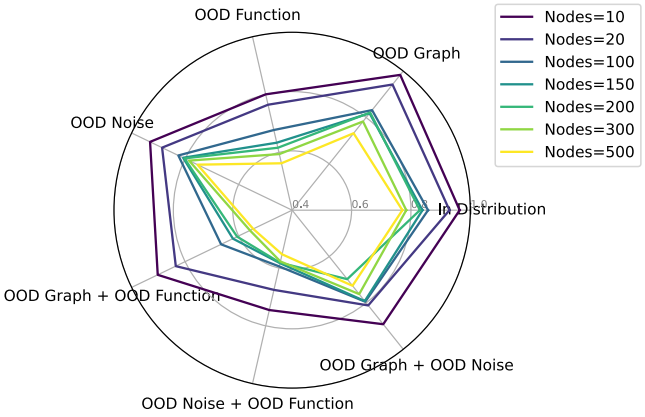

Figure 4: Generalization property of `CauScale` on OOD graphs, noise, and mechanism functions.

this study to a subset of the training set with node counts $n \in \{10, 20, 100\}$. Figure 3 illustrates the results averaged across all causal mechanisms. The results show the advantage of our implemented components. First, the tied-attention mechanism demonstrates superior computational efficiency, achieving an inference speed six times faster than vanilla attention on graphs with ($n = 100, |E| = 400$). It also yields higher mean average precision score, striking a perfect balance between efficiency and accuracy. Second, our pairwise processing head leads to a much more lower degree of cyclicity (0.0%) compared to the vanilla prediction head used in AVICI (0.0-0.25%).

Pooling strategy of reduction unit

We conduct a lightweight ablation study using graphs with node counts of $n = \{10, 20, 50\}$ during training and testing to efficiently evaluate different pooling strategies within the *reduction unit*. In addition to the average pooling employed in `CauScale`, we compare two alternative downsampling techniques: strided pooling and max pooling. Table 1 demonstrates that average pooling consistently achieves superior performance across experimented graph sizes.

8.2 Generalization Analysis

We further evaluate the generalization capability of the synthetic-data-trained model described in our main experiment across OOD graph structures generated by Stochastic Block Models, OOD noise distributions (uniform and Laplace), and OOD functions (sigmoid and polynomial). Figure 4 reveals that the model demonstrates strong generalization to OOD graph structures but exhibits greater sensitivity to OOD noise patterns and mechanism functions. These findings indicate that future model training should prioritize generating datasets with more diverse noise distributions and mechanism functions to enhance robustness.

8.3 Sample Size analysis

We conduct a sample size analysis on our trained models in our main experiment, with inference sample sizes varying from 500 to 4000 for synthetic data and from 1000 to 20000 for SERGIO-GRN data. Figure 5 shows that the model trained on synthetic data achieves peak performance with sample size of 2000, while the model for SERGIO-GRN achieves the best performance with a sample size of 20000. This suggests that more complex causal mechanisms necessitate larger sample sizes for accurate inference.

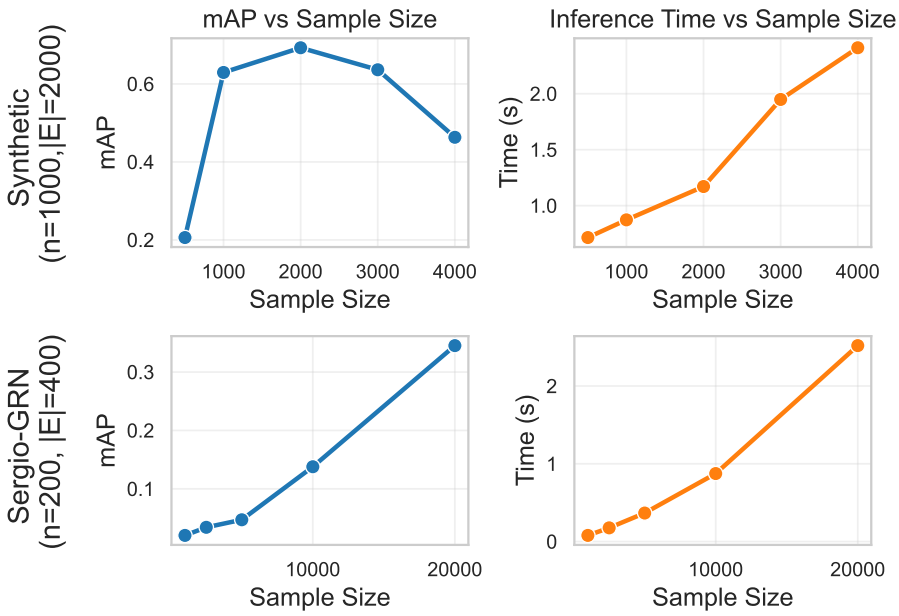

Figure 5: Sample size analysis by mean average precision.

References

Brouillard, P., Lachapelle, S., Lacoste, A., Lacoste-Julien, S., and Drouin, A. (2020). Differentiable causal discovery from interventional data. *Advances in Neural Information Processing Systems*, 33:21865–21877.

Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554.

Chu, D., Zabet, N. R., and Mitavskiy, B. (2009). Models of transcription factor binding: sensitivity of activation functions to model assumptions. *Journal of Theoretical Biology*, 257(3):419–429.

Dhir, A., Ashman, M., Requeima, J., and van der Wilk, M. (2025). A meta-learning approach to bayesian causal discovery. In *The Thirteenth International Conference on Learning Representations*.

Dibaeinia, P. and Sinha, S. (2020). Sergio: a single-cell expression simulator guided by gene regulatory networks. *Cell systems*, 11(3):252–271.

Goudet, O., Kalainathan, D., Caillou, P., Guyon, I., Lopez-Paz, D., and Sebag, M. (2017). Causal generative neural networks. *arXiv preprint arXiv:1711.08936*.

Hauser, A. and Bühlmann, P. (2012). Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. *The Journal of Machine Learning Research*, 13(1):2409–2464.

Hyvärinen, A., Hurri, J., and Hoyer, P. O. (2001). Independent component analysis. In *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision*, pages 151–175. Springer.

Ke, N. R., Bilaniuk, O., Goyal, A., Bauer, S., Larochelle, H., Schölkopf, B., Mozer, M. C., Pal, C., and Bengio, Y. (2019). Learning neural causal models from unknown interventions. *arXiv preprint arXiv:1910.01075*.

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 *International Conference on Learning Representations*.

Lachapelle, S., Brouillard, P., Deleu, T., and Lacoste-Julien, S. (2019). Gradient-based neural dag learning. *arXiv preprint arXiv:1906.02226*.

Lippe, P., Cohen, T., and Gavves, E. (2021). Efficient neural causal discovery without acyclicity constraints. *arXiv preprint arXiv:2107.10483*.

Lorch, L., Sussex, S., Rothfuss, J., Krause, A., and Schölkopf, B. (2022). Amortized inference for causal structure learning. *Advances in Neural Information Processing Systems*, 35.

Nazaret, A., Hong, J., Azizi, E., and Blei, D. (2023). Stable differentiable causal discovery. In *Forty-first International Conference on Machine Learning*.

Rao, R., Liu, J., Verkuil, R., Meier, J., Canny, J. F., Abbeel, P., Sercu, T., and Rives, A. (2021). Msa transformer. *bioRxiv*.

Sanchez, P., Liu, X., O'Neil, A. Q., and Tsaftaris, S. A. (2023). Diffusion models for causal discovery via topological ordering. In *The Eleventh International Conference on Learning Representations*.

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

Spirtes, P., Glymour, C., and Scheines, R. (2000). Causation, prediction, and search. adaptive computation and machine learning series. *The MIT Press*, 49:77–78.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78.

Wu, M., Bao, Y., Barzilay, R., and Jaakkola, T. S. (2025). Sample, estimate, aggregate: A recipe for causal discovery foundation models. *Trans. Mach. Learn. Res.*, 2025.

Zheng, X., Aragam, B., Ravikumar, P., and Xing, E. P. (2018). DAGs with NO TEARS: Continuous Optimization for Structure Learning. In *Advances in Neural Information Processing Systems*.

Zhu, S., Ng, I., and Chen, Z. (2019). Causal discovery with reinforcement learning. *arXiv preprint arXiv:1906.04477*.