DIFFUSION MODELS FOR CAUSAL DISCOVERY VIA TOPOLOGICAL ORDERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Discovering causal relations from observational data becomes possible with additional assumptions such as considering the functional relations to be constrained as nonlinear with additive noise. In this case, the *Hessian* of the data log-likelihood can be used for finding leaf nodes in a causal graph. This observation results in an iterative algorithm for *topological ordering* which sequentially finds and removes leaf nodes in a graph. Topological ordering approaches for causal discovery are efficient because they reduce the search from graph space to a permutation space. Inspired by recent innovations in diffusion probabilistic models (DPMs), we propose *DiffAN*, a topological ordering algorithm that leverages DPMs for computing the *Hessian*. In particular, we update the learned Hessian without re-training the neural network and perform ordering over a batch, which allows scaling to datasets with more samples and variables. We show empirically that our method scales exceptionally well to datasets with up to 500 nodes and up to 10^5 samples while still performing on par over small datasets with state-of-the-art causal discovery methods.

1 INTRODUCTION

Understanding the causal structure of a problem is important for areas such as economics, biology (Sachs et al., 2005) and healthcare (Sanchez et al., 2022), especially when reasoning about the effect of interventions. When interventional data from randomised trials are not available, causal discovery methods (Glymour et al., 2019) may be employed to discover the causal structure of a problem solely from observational data. Causal structure is typically modelled as a directed acyclic graph (DAG) $\mathcal{G}$ in which each node is associated with a random variable and each edge represents a causal mechanism i.e. how one variable influences another.

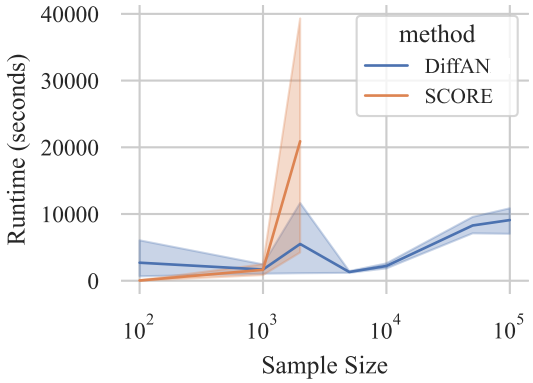

Figure 1: Plot showing run time in seconds for different sample sizes, for discovery of causal graphs with 500 nodes. Most causal discovery methods have prohibitive run time and memory cost for datasets with many samples; the previous state-of-the-art SCORE algorithm (Rolland et al., 2022) which is included in this graph cannot be computed beyond 2000 samples in a machine with 64GB of RAM. By contrast, our method DiffAN has a reasonable run time even for numbers of samples two orders of magnitude larger than capable by most existing methods.

However, learning such a model from data is NP-hard (Chickering, 1996). Traditional methods search the DAG space by testing for conditional independence between variables (Spirtes et al., 1993) or by optimising some goodness of fit measure (Chickering, 2002). Unfortunately, solving the search problem with a greedy combinatorial optimisation method can be expensive and does not scale to high-dimensional problems.

In line with previous work (Teyssier & Koller, 2005; Park & Klabjan, 2017; Bühlmann et al., 2014; Solus et al., 2021; Wang et al., 2021; Rolland et al., 2022), we can speed up the combinatorial search

problem over the space of DAGs by rephrasing it as a *topological ordering* task, ordering from leaf nodes to root nodes. The search space over DAGs with d nodes and $(d^2 - d)/2$ possible edges is much larger than the space of permutations over d variables. Once a topological ordering of the nodes is found, the potential causal relations between later (cause) and earlier (effect) nodes can be pruned with a feature selection algorithm (e.g. Bühlmann et al. (2014)) to yield a graph which is naturally directed and acyclic without further optimisation.

Recently, Rolland et al. (2022) proposed the SCORE algorithm which uses the Hessian of the data log-likelihood, $\nabla^2_{\mathrm{x}} \log p(\mathbf{x})$ (i.e. the Jacobian of the *score* [1]) for topological ordering. By verifying which elements of $\nabla^2_{\mathrm{x}} \log p(\mathbf{x})$'s diagonal are constant across all data points, leaf nodes can be iteratively identified and removed. The procedure from Rolland et al. (2022) scales linearly with the number of variables d but with polynomial complexity with the number of samples n i.e. $O(dn^3)$, due to the Stein gradient estimator (Li & Turner, 2018), which hinders its use for high-dimensional causal discovery.

Here, we use neural networks (NN) trained with diffusion probabilistic modeling (DPM) for learning the score function $\nabla_{\mathrm{x}} \log p(\mathbf{x})$, inspired by Rolland et al. (2022). We propose a method to infer an additive residue which allows computing the score at each leaf-removal step *without* re-training the NN. In addition, the NN is trained over the entire dataset (n samples) but only a subsampled batch is used for finding leaf nodes.Thus, once the score model is learned, we can use it to order the graph with constant complexity on n, enabling causal discovery in high-dimensional settings. Interestingly, our algorithm does not require architectural constraints on the neural network, as in previous causal discovery methods based on neural networks (Lachapelle et al., 2020; Zheng et al., 2020; Yu et al., 2019; Ng et al., 2022). Our training procedure does not learn the causal mechanism directly, but the score of the data distribution.

Contributions. In summary, we propose DiffAN, an identifiable algorithm leveraging a diffusion probabilistic model for topological ordering that enables causal discovery assuming an additive noise model: (i) To the best of our knowledge, we present the first causal discovery algorithm based on denoising diffusion training. The score estimated with the diffusion model is used to find leaf nodes iteratively; (ii) We prove (Section 2.2) that ANMs are identifiable from the variance of the log-likelihood's Hessian for a more general family of noise distribution, extending identifiability statements from Rolland et al. (2022); (iii) We show in Section 3 that when a leaf node is removed, the score of the new distribution can be estimated from the original score (before leaf removal) and its Jacobian. It allow us to propose an efficient algorithm for topological ordering which does not depend on re-training the score after each leaf removal iteration; (iv) We estimate the second-order derivatives (score's Jacobian or Hessian) of a data distribution using neural networks with diffusion training via backpropagation; (v) We show empirically that our method performs on a par with state-of-the-art causal discovery methods in small graph datasets but scales better to bigger datasets with more nodes and samples.

2 PRELIMINARIES

2.1 PROBLEM DEFINITION

We consider the problem of discovering the causal structure between d variables, given a probability distribution $p(\mathbf{x})$ from which a d-dimensional random vector $\mathbf{x} = (\mathrm{x}_1, \ldots, \mathrm{x}_d)$ can be sampled. We assume that the true causal structure is described by a DAG $\mathcal{G}$ containing d nodes. Each node represents a random variable x_i and edges represent the presence of causal relations between them. In other words, we can say that $\mathcal{G}$ defines a structural causal model (SCM) consisting of a collection of assignments $\mathrm{x}_i := f_i(Pa(\mathrm{x}_i), \epsilon_i)$, where $Pa(\mathrm{x}_i)$ are the parents of x_i in $\mathcal{G}$, and ϵ_i is a noise term independent of x_i, also called exogenous noise. ϵ_i are i.i.d. from a smooth distribution p^{ϵ}. The SCM entails a unique distribution $p(\mathbf{x}) = \prod_{i=1}^{d} p(\mathrm{x}_i \mid Pa(\mathrm{x}_i))$ over the variables $\mathbf{x}$ (Peters et al., 2017). The observational input data are $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, where n is number of samples. The target output is an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{d \times d}$.

[1] Note that, throughout the paper, we use the term "score" to denote the gradient of the data log-likelihood $\nabla_{\mathrm{x}} \log p(\mathbf{x})$; this is different to causal literature which uses "score" to refer to the optimisation function in combinatorial search algorithms.

The **topological ordering** (also called *causal ordering* or *causal list*) of a DAG $\mathcal{G}$ is defined as a non-unique permutation π of d nodes such that a given node always appears first in the list than its descendants. Formally, $\pi_i < \pi_j \iff j \in De_{\mathcal{G}}(\mathrm{x}_i)$ where $De_{\mathcal{G}}(\mathrm{x}_i)$ are the descendants of the ith node in $\mathcal{G}$ (Appendix B in Peters et al. (2017)).

2.2 Nonlinear Additive Noise Models

Learning a unique $\boldsymbol{A}$ from $\boldsymbol{X}$ with observational data requires additional assumptions. A common class of methods called additive noise models (ANM) (Shimizu et al., 2006; Hoyer et al., 2008; Peters et al., 2014; Bühlmann et al., 2014) explores asymmetries in the data by imposing functional assumptions on the data generation process. In most cases, they assume that assignments take the form $\mathrm{x}_i := f_i(Pa(\mathrm{x}_i)) + \epsilon_i$ with $\epsilon_i \sim p^\epsilon$. Here we focus on the case described by Peters et al. (2014) where f_i is nonlinear. We use the notation f_i for $f_i(Pa(\mathrm{x}_i))$ because the arguments of f_i will always be $Pa(\mathrm{x}_i)$ throughout this paper. We highlight that f_i does not depend on i.

Identifiability. We assume that the SCM follows an additive noise model (ANM) which is known to be identifiable from observational data (Hoyer et al., 2008; Peters et al., 2014). We also assume causal sufficiency, i.e. there are no hidden variables that are a common cause of at least two observed variables. In addition, corollary 33 from Peters et al. (2014) states that the true topological ordering of the DAG, as in our setting, is identifiable from a $p(\mathbf{x})$ generated by an ANM without requiring causal minimality assumptions.

Finding Leaves with the Score. Rolland et al. (2022) propose that the score of an ANM with distribution $p(\mathbf{x})$ can be used to find leaves[1]. Before presenting how to find the leaves, we derive, following Lemma 2 in Rolland et al. (2022), an analytical expression for the score which can be written as

$$\begin{aligned}
\nabla_{\mathrm{x}_j} \log p(\mathbf{x}) &= \nabla_{\mathrm{x}_j} \log \prod_{i=1}^{d} p(\mathrm{x}_i \mid Pa(\mathrm{x}_i)) \\
&= \nabla_{\mathrm{x}_j} \sum_{i=1}^{d} \log p(\mathrm{x}_i \mid Pa(\mathrm{x}_i)) \\
&= \nabla_{\mathrm{x}_j} \sum_{i=1}^{d} \log p^\epsilon\left(\mathrm{x}_i - f_i\right) && \triangleright \text{ Using } \epsilon_i = \mathrm{x}_i - f_i \\
&= \frac{\partial \log p^\epsilon\left(\mathrm{x}_j - f_j\right)}{\partial \mathrm{x}_j} - \sum_{i \in Ch(\mathrm{x}_j)} \frac{\partial f_i}{\partial \mathrm{x}_j} \frac{\partial \log p^\epsilon\left(\mathrm{x}_i - f_i\right)}{\partial x}.
\end{aligned} \tag{1}$$

Where $Ch(\mathrm{x}_j)$ denotes the children of x_j. Note that if x_j does not have a child node, the second term on the last row disappears. In this case, the Jacobian of the score $\nabla^2_{\mathrm{x}_j} \log p(\mathbf{x})$ (also denoted as the Hessian $\boldsymbol{H}_{j,j}(\log p(\mathbf{x}))$[2]) becomes a constant. This observation provides a mechanism to identify leaf nodes in a graph because if the second-order derivative is constant, the variance across all data points should be zero.

Lemma 1. *Given a nonlinear ANM with a noise distribution p^ϵ,*

$$\mathrm{Var}_{\boldsymbol{X}}\left[\boldsymbol{H}_{j,j}(\log p(\mathbf{x}))\right] = 0 \iff j \text{ is a leaf } \;\&\; \frac{d \log p^\epsilon}{dx^2} = a, \tag{2}$$

where a is a constant and $\boldsymbol{H}(\log p(\mathbf{x})) \in \mathbb{R}^{d,d}$ is the Hessian matrix of the log-density $\log p(\mathbf{x})$. Therefore, the leaves are only identifiable using the variance of the log-likelihood's Hessian for ANMs with noise distribution that have constant second-order derivatives.

See proof in Appendix A.1. Note that Lemma 1 is a generalisation of Rolland et al. (2022) which only prove that Equation 2 can identify leaves in nonlinear ANMs with Gaussian noise. We derive a more general formulation which additionally requires the second-order derivative of the noise distribution to be constant. Indeed, the condition $\frac{d \log p^\epsilon}{dx^2} = a$ is true for p^ϵ following a Gaussian distribution which is consistent with Rolland et al. (2022).

[1] We refer to nodes without children in a DAG $\mathcal{G}$ as leaves.

[2] The subscripts i, j in $\boldsymbol{H}_{i,j}$ denote an element at the ith row and jth column.

2.3 DIFFUSION MODELS APPROXIMATE THE SCORE

The process of learning to denoise (Vincent, 2011) can approximate that of matching the score (Hyvärinen, 2005). A diffusion process gradually adds noise to a data distribution over time. Diffusion probabilistic models (DPMs) Sohl-Dickstein et al. (2015); Ho et al. (2020); Song et al. (2021) learn to reverse the diffusion process, starting with noise and recovering the data distribution. The diffusion process gradually adds Gaussian noise, with a time-dependent variance α_t, to a sample $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$ from the data distribution. Thus, the noisy variable $\mathbf{x}_t$, with $t \in [0, T]$, is learned to correspond to versions of $\mathbf{x}_0$ perturbed by Gaussian noise following $p(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\boldsymbol{I}\right)$, where $\alpha_t := \prod_{j=0}^{t}(1-\beta_j)$, β_j is the variance scheduled between $[\beta_{\text{min}}, \beta_{\text{max}}]$ and $\boldsymbol{I}$ is the identity matrix. DPMs (Ho et al., 2020) are learned with a weighted sum of denoising score matching objectives at different perturbation scales with

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\mathbf{x}_0, t, \epsilon}\left[\lambda(t)\left\|\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \epsilon\right\|_2^2\right], \tag{3}$$

where $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon$, with $\mathbf{x}_0 \sim p(\mathbf{x})$ being a sample from the data distribution, $t \sim \mathcal{U}(0, T)$ and $\epsilon \sim \mathcal{N}(0, \boldsymbol{I})$ is the noise. $\lambda(t)$ is a loss weighting term following Ho et al. (2020).

Throughout this paper, the important observation is that the trained model $\boldsymbol{\epsilon}_\theta$ approximates the score $\nabla_{\mathrm{x}_j} \log p(\mathbf{x})$ of the data (Song & Ermon, 2019). $\boldsymbol{\epsilon}_\theta : \mathbb{R}^{d+1} \to \mathbb{R}^d$ is a neural network with learnable parameters θ. The input is $d+1$ variables because we concatenate the t scalar variable with the input data.

3 THE DECIDUOUS SCORE

Discovering the complete topological ordering with the distribution's Hessian (Rolland et al., 2022) is done by finding the leaf node (Equation 2), appending the leaf node x_l to the ordering list π and removing the data column corresponding to x_l from $\boldsymbol{X}$ before the next iteration $d-1$ times. Rolland et al. (2022) estimate the score's Jacobian (Hessian) at each iteration.

Instead, we explore an alternative approach that does not require estimation of a new score after each leaf removal. In particular, we describe how to adjust the score of a distribution after each leaf removal, terming this a "deciduous score"[1]. We obtain an analytical expression for the deciduous score and derive a way of computing it, based on the original score before leaf removal. In this section, we only consider that $p(\mathbf{x})$ follows a distribution described by an ANM, we pose no additional assumptions over the noise distribution.

Definition 1. *Considering a DAG $\mathcal{G}$ which entails a distribution $p(\mathbf{x}) = \prod_{i=1}^{d} p(\mathrm{x}_i \mid Pa(\mathrm{x}_i)) \in \mathbb{R}^d$. Let $p(\mathbf{x}_{-l}) = \frac{p(\mathbf{x})}{p(\mathrm{x}_l \mid Pa(\mathrm{x}_l))} \in \mathbb{R}^{d-1}$ be $p(\mathbf{x})$ without the random variable corresponding to the leaf node x_l. The* deciduous *score $\nabla \log p(\mathbf{x}_{-l}) \in \mathbb{R}^{d-1}$ is the score of the distribution $p(\mathbf{x}_{-l})$.*

Lemma 2. *Given a ANM which entails a distribution $p(\mathbf{x})$, we can use Equation 1 to find an analytical expression for an additive residue Δ_l between the distribution's score $\nabla \log p(\mathbf{x})$ and its deciduous score $\nabla \log p(\mathbf{x}_{-l})$ such that*

$$\Delta_l = \nabla \log p(\mathbf{x}) - \nabla \log p(\mathbf{x}_{-l}). \tag{4}$$

In particular, Δ_l is a vector $\{\delta_j \mid \forall j \in [1, \ldots, d] \setminus l\}$ where the residue w.r.t. a node x_j can be denoted as

$$\begin{aligned} \delta_j &= \nabla_{\mathrm{x}_j} \log p(\mathbf{x}) - \nabla_{\mathrm{x}_j} \log p(\mathbf{x}_{-l}) \\ &= -\frac{\partial f_i}{\partial \mathrm{x}_j} \frac{\partial \log p^{\epsilon}(\mathrm{x}_i - f_i)}{\partial x}. \end{aligned} \tag{5}$$

If $\mathrm{x}_j \notin Pa(\mathrm{x}_l)$, $\delta_j = 0$.

Proof. Observing Equation 1, the score $\nabla_{\mathrm{x}_j} \log p(\mathbf{x})$ only depends on the following random variables (i) $Pa(\mathrm{x}_j)$, (ii) $Ch(\mathrm{x}_j)$, and (iii) $Pa(Ch(\mathrm{x}_j))$. We consider x_l to be a leaf node, therefore $\nabla_{\mathrm{x}_j} \log p(\mathbf{x})$ only depends on x_l if $\mathrm{x}_j \in Pa(\mathrm{x}_l)$. If $\mathrm{x}_j \in Pa(\mathrm{x}_l)$, the only term depending on $\nabla_{\mathrm{x}_j} \log p(\mathbf{x})$ dependent on x_l is one of the terms inside the summation. □

[1] An analogy to *deciduous* trees which seasonally shed leaves during autumn.

However, we wish to estimate the deciduous score $\nabla \log p(\mathbf{x}_{-l})$ without direct access to the function f_l, to its derivative, nor to the distribution p^{ϵ}. Therefore, we now derive an expression for Δ_l using solely the score and the Hessian of $\log p(\mathbf{x})$.

Theorem 1. *Consider an ANM of distribution $p(\mathbf{x})$ with score $\nabla \log p(\mathbf{x})$ and the score's Jacobian $\boldsymbol{H}(\log p(\mathbf{x}))$. The additive residue Δ_l necessary for computing the deciduous score (as in Proposition 2) can be estimated with*

$$\Delta_l = \boldsymbol{H}_l(\log p(\mathbf{x})) \cdot \frac{\nabla_{\mathbf{x}_l} \log p(\mathbf{x})}{\boldsymbol{H}_{l,l}(\log p(\mathbf{x}))}. \tag{6}$$

See proof in Appendix A.2.

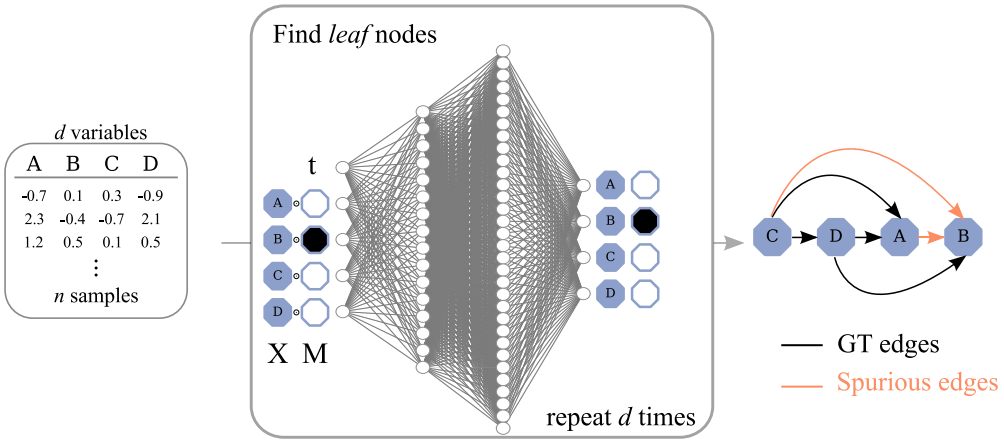

Figure 2: Topological ordering with diffusion models by iteratively finding leaf nodes. At each iteration, one leaf node is found using Equation 9. In the subsequent iteration, the previous leaves are removed, reducing the search space. After topological ordering (as illustrated on the right side), the presence of edges (causal mechanisms) between variables can be inferred such that parents of each variable are selected from the preceding variables in the ordered list. Spurious edges can be pruned with feature selection as a post-processing step (Bühlmann et al., 2014; Lachapelle et al., 2020; Rolland et al., 2022).

4 Causal Discovery with Diffusion Models

DPMs approximate the score of the data distribution (Song & Ermon, 2019). In this section, we explore how to use DPMs to perform leaf discovery and compute the deciduous score, based on Theorem 1, for iteratively finding and removing leaf nodes *without* re-training the score.

4.1 Approximating the Score's Jacobian via Diffusion Training

The score's Jacobian can be approximated by learning the score $\boldsymbol{\epsilon}_\theta$ with denoising diffusion training of neural networks and back-propagating (Rumelhart et al., 1986)[1] from the output to the input variables. It can be written, for an input data point $\boldsymbol{x} \in \mathbb{R}^d$, as

$$\boldsymbol{H}_{i,j} \log p(\boldsymbol{x}) \approx \nabla_{i,j} \boldsymbol{\epsilon}_\theta(\boldsymbol{x}, t), \tag{7}$$

where $\nabla_{i,j} \boldsymbol{\epsilon}_\theta(\boldsymbol{x}, t)$ means the ith output of $\boldsymbol{\epsilon}_\theta$ is backpropagated to the jth input. The diagonal of the Hessian in Equation 7 can, then, be used for finding leaf nodes as in Equation 2.

In a two variable setting, it is sufficient for causal discovery to (i) train a diffusion model (Equation 3); (ii) approximate the score's Jacobian via backpropagation (Equation 7); (iii) compute variance of

[1]The Jacobian of a neural network can be efficiently computed with auto-differentiation libraries such as functorch (Horace He, 2021).

the diagonal across all data points; (iv) identify the variable with lowest variance as effect (Equation 2). We illustrate in Appendix C the Hessian of a two variable SCM computed with a diffusion model.

4.2 Topological Ordering

When a DAG contains more than two nodes, the process of finding leaf nodes (i.e. the topological order) needs to be done iteratively. The naive (greedy) approach would be to remove the leaf node from the dataset, recompute the score, and compute the variance of the new distribution's Hessian to identify the next leaf node (Rolland et al., 2022). Since we employ diffusion models to estimate the score, this equates to re-training the model each time after a leaf is removed.

We hence propose a method to compute the deciduous score $\nabla \log p(\mathbf{x}_{-l})$ using Theorem 1 to remove leaves from the initial score *without* re-training the neural network. In particular, assuming that a leaf x_l is found, the residue Δ_l can be approximated[1] with

$$\Delta_l(\boldsymbol{x}, t) \approx \nabla_l \boldsymbol{\epsilon}_\theta(\boldsymbol{x}, t) \cdot \frac{\boldsymbol{\epsilon}_\theta(\boldsymbol{x}, t)_l}{\nabla_{l,l} \boldsymbol{\epsilon}_\theta(\boldsymbol{x}, t)} \tag{8}$$

where $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}, t)_l$ is output corresponding to the leaf node. Note that the term $\nabla_l \boldsymbol{\epsilon}_\theta(\boldsymbol{x}, t)$ is a vector of size d and the other term is a scalar. During topological ordering, we compute Δ_π, which is the summation of Δ_l over all leaves already discovered and appended to π. Naturally, we only compute Δ_l w.r.t. nodes $\mathrm{x}_j \notin \pi$ because $\mathrm{x}_j \in \pi$ have already been ordered and are not taken into account anymore.

In practice, we observe that training $\boldsymbol{\epsilon}_\theta$ on $\boldsymbol{X}$ but using a subsample $\boldsymbol{B} \in \mathbb{R}^{k \times d}$ of size k randomly sampled from $\boldsymbol{X}$ increases speed without compromising performance (see Section 4.3). In addition, we observe that the absolute value of the residue δ_l decreases if the values of x_l are set to zero once the leaf node is discovered. Therefore, we apply a mask $\boldsymbol{M}_\pi \in \{0, 1\}^{k \times d}$ over leaves discovered in the previous iterations and compute only the Jacobian of the outputs corresponding to $\mathbf{x}_{-l}$. $\boldsymbol{M}_\pi$ is updated after each iteration based on the ordered nodes π. We then find a leaf node according to

$$\text{leaf} = \underset{\mathrm{x}_i \in \mathbf{x}}{\arg\min} \operatorname{Var}_{\boldsymbol{B}} \left[\nabla_{\mathbf{x}} \left(score(\boldsymbol{M}_\pi \odot \boldsymbol{B}, t) \right) \right], \tag{9}$$

where $\boldsymbol{\epsilon}_\theta$ is a DPM trained with Equation 3. See Appendix E.2 for the choice of t. This topological ordering procedure is formally described in Algorithm 1, $score(-\pi)$ means that we only consider the outputs for nodes $\mathrm{x}_j \notin \pi$

Algorithm 1: Topological Ordering with DiffAN

Input: $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, trained diffusion model $\boldsymbol{\epsilon}_\theta$, ordering batch size k
$\pi = []$, $\Delta_\pi = \mathbf{0}^{k \times d}$, $\boldsymbol{M}_\pi = \mathbf{1}^{k \times d}$, $score = \boldsymbol{\epsilon}_\theta$
while $\|\pi\| \neq d$ **do**
  $\boldsymbol{B} \overset{k}{\leftarrow} \boldsymbol{X}$ // Randomly sample a batch of k elements
  $\boldsymbol{B} \leftarrow \boldsymbol{B} \circ \boldsymbol{M}_\pi$ // Mask removed leaves
  $\Delta_\pi = \text{Get}\Delta_\pi(score, \boldsymbol{B})$ // Sum of Equation 8 over π
  $score = score(-\pi) + \Delta_\pi$ // Update score with residue
  $leaf = \text{GetLeaf}(score, \boldsymbol{B})$ // Equation 9
  $\pi = [leaf, \pi]$ // Append leaf to ordered list
  $\boldsymbol{M}_{:,leaf} = \mathbf{0}$ // Set discovered leaf to zero
end
Output: Topological order π

4.3 Computational Complexity and Practical Considerations

We now study the complexity of topological ordering with DiffAN w.r.t. the number of samples n and number of variables d in a dataset. In addition, we discuss what are the complexities of a greedy version as well as approximation which only utilises masking.

[1]The diffusion model itself is an approximation of the score, therefore its gradients are approximations of the score derivatives.

Complexity on n. Our method separates learning the score $\boldsymbol{\epsilon}_\theta$ from computing the variance of the Hessian's diagonal across data points, in contrast to Rolland et al. (2022). We use all n samples in $\boldsymbol{X}$ for learning the score function with diffusion training (Equation 3). It does *not* involve expensive constrained optimisation techniques[1] and we train the model for a fixed number of epochs (which is linear with n) or until reaching the early stopping criteria. We use a MLP that grows in width with d but it does not significantly affect complexity. Therefore, we consider training to be $O(n)$. Moreover, Algorithm 1 is computed over a batch $\boldsymbol{B}$ with size $k < n$ instead of the entire dataset $\boldsymbol{X}$, as described in Equation 9. Note that the number of samples k in $\boldsymbol{B}$ can be arbitrarily small and constant for different datasets. In Section 5.2, we verify that the accuracy of causal discovery initially improves as k is increased but eventually tapers off.

Complexity on d. Once $\boldsymbol{\epsilon}_\theta$ is trained, a topological ordering can be obtained by running $\nabla_{\boldsymbol{x}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}, t)$ d times. Moreover, computing the Jacobian of the score requires back-propagating the gradients $d - i$ times, where i is the number of nodes already ordered in a given iteration. Finally, computing the deciduous score's residue (Equation 8) means computing gradient of the i nodes. Resulting in a complexity of $O(d \cdot (d - i) \cdot i)$ with i varying from 0 to d which can be described by $O(d^3)$. The final topological ordering complexity is therefore $O(n + d^3)$.

DiffAN Greedy. A greedy version of the algorithm re-trains the $\boldsymbol{\epsilon}_\theta$ after each leaf removal iteration. In this case, the deciduous score is not computed, decreasing the complexity w.r.t. d but increasing w.r.t. n. DiffAN greedy has $O(nd^2)$ ordering complexity.

DiffAN Masking. We verify empirically that the masking procedure described in Section 4.2 can significantly reduce the deciduous score's residue absolute value while maintaining causal discovery capabilities. In DiffAN Masking, we do not re-train the $\boldsymbol{\epsilon}_\theta$ nor compute the deciduous score. This ordering algorithm is an approximation but has shown to work well in practice while showing remarkable scalability. DiffAN Masking has $O(n + d^2)$ ordering complexity.

5 Experiments

In our experiments, we train a NN with a DPM objective to perform topological ordering and follow this with a pruning post-processing step (Bühlmann et al., 2014). The performance is evaluated on synthetic and real data and compared to state-of-the-art causal discovery methods from observational data which are either ordering-based or gradient-based methods, **NN architecture.** We use a 4-layer multilayer perceptron (MLP) with LeakyReLU and layer normalisation. **Metrics.** We use the structural Hamming distance (SHD), Structural Intervention Distance (SID) (Peters & Bühlmann, 2015), *Order Divergence* (Rolland et al., 2022) and run time in seconds. See Appendix D.3 for details of each metric. **Baselines.** We use CAM (Bühlmann et al., 2014), GranDAG (Lachapelle et al., 2020) and SCORE (Rolland et al., 2022). We apply the pruning procedure of Bühlmann et al. (2014) to all methods. See detailed results in Appendix D.

5.1 Synthetic Data

In this experiment, we consider causal relationships with f_i being a function sampled from a Gaussian Process (GP) with radial basis function kernel of bandwidth one. We generate data from additive noise models which follow a Gaussian, Exponential or Laplace distributions with noise scales in the intervals $\{[0.4, 0.8], [0.8, 1.2], [1, 1]\}$, which are known to be identifiable (Peters et al., 2014). The causal graph is generated using the Erdös-Rényi (ER) (Erdős et al., 1960) and Scale Free (SF) (Bollobás et al., 2003) models. For a fixed number of nodes d, we vary the sparsity of the sampled graph by setting the average number of edges to be either d or $5d$. We use the notation $[d]$[graph type][sparsity] for indicating experiments over different synthetic datasets. We show that DiffAN performs on par with baselines while being extremely fast. We also explore the role of overfitting in Appendix E.1, the difference between DiffAN with masking only and the greedy version in Appendix E.3, how we choose t during ordering in Appendix E.2 and we give results stratified by experiment in Appendix E.4.

[1] Such as the Augmented Lagrangian method (Zheng et al., 2018; Lachapelle et al., 2020).

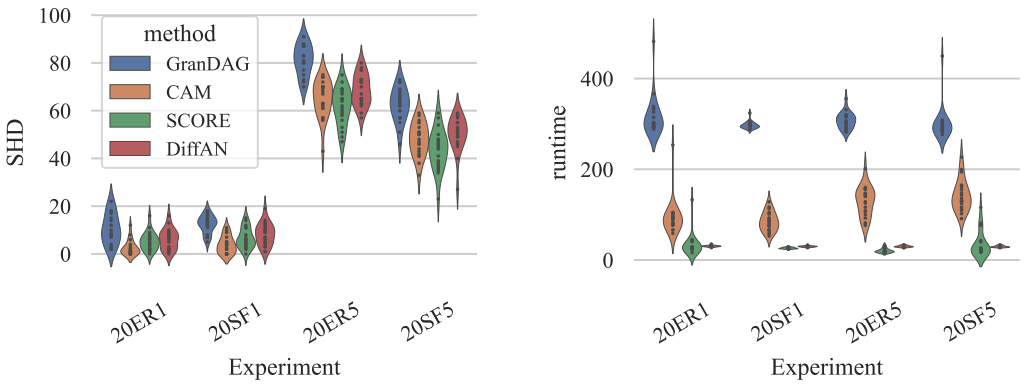

Figure 3: SHD (left) and run time in seconds (right) for experiments of synthetic data graphs for graphs with 20 nodes. The variation in the violinplots come from 3 different seeds over dataset generated from 3 different noise type and 3 different noise scales. Therefore, we run a total of 27 experiments for each method and synthetic datasets type

5.2 Scaling up with DiffAN Masking

We now verify how DiffAN scales to bigger datasets, in terms of the number of samples n. Here, we use *DiffAN masking* because computing the residue with DiffAN would be too expensive for very big d. We evaluate only the topological ordering, ignoring the final pruning step. Therefore, the performance will be measured solely with the Order Divergence metric. We evaluate how DiffAN compares to SCORE (Rolland et al., 2022), the previous state-of-the-art, in terms of run time (in seconds) and and the performance (order divergence) over a dataset with $d = 500$ and different sample sizes $n \in 10^2, \dots, 10^5$. As illustrated in Figure 1 and Figure 4 (left), SCORE is faster for a small sample size but as the size of the dataset increases, DiffAN becomes the more tractable option.

Ordering batch size. An important aspect of our method, discussed in Section 4.3, that allows scalability in terms of n is the separation between learning the score function $\boldsymbol{\epsilon}_\theta$ and computing the Hessian variance across a batch of size k, with $k < n$. Therefore, we show empirically, as illustrated in Figure 4 (right), that decreasing k does not strongly impact performance for datasets with $d \in 10, 20, 50$.

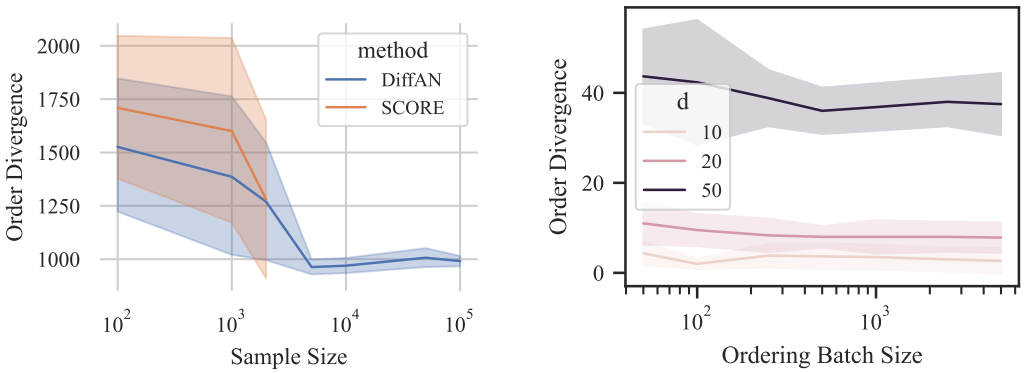

Figure 4: Accuracy of DiffAN as the dataset size is scaled up in numbers of data samples n (left) and as the batch size for computing the Hessian variance is changed (right).

5.3 Real Data

We consider two real datasets: (i) Sachs: A protein signaling network based on expression levels of proteins and phospholipids (Sachs et al., 2005). We consider only the observational data ($n = 853$ samples) since our method targets discovery of causal mechanisms when only observational data is

available. The ground truth causal graph given by Sachs et al. (2005) has 11 nodes and 17 edges. (ii) SynTReN: We also evaluate the models on a pseudo-real dataset sampled from SynTReN generator (Van den Bulcke et al., 2006). Results, in Table 1, show that our method is competitive against other state-of-the-art causal discovery baselines on real datasets.

Table 1: SHD and SID results over real datasets.

	Sachs		SynTReN	
	SHD	SID	SHD	SID
CAM	12	55	40.5	152.3
GraN-DAG	13	47	34.0	161.7
SCORE	12	45	36.2	193.4
DiffAN (ours)	13	56	39.7	173.5

6 Related Works

The observation that a causal DAG can be partially represented with a topological ordering goes back to Verma & Pearl (1990). Searching the topological ordering space instead of searching over the space of DAGs has been done with greedy Markov Chain Monte Carlo (MCMC) (Friedman & Koller, 2003), greedy hill-climbing search (Teyssier & Koller, 2005), arc search (Park & Klabjan, 2017), restricted maximum likelihood estimators (Bühlmann et al., 2014), sparsest permutation (Raskutti & Uhler, 2018; Lam et al., 2022; Solus et al., 2021), and reinforcement learning (Wang et al., 2021). In linear additive models, Ghoshal & Honorio (2018); Chen et al. (2019) propose an approach, under some assumptions on the noise variances, to discover the causal graph by sequentially identifying leaves based on an estimation of the precision matrix.

7 Conclusion

We have presented the first method using diffusion models for causal discovery. Since DPMs approximate the score of the data distribution, they can be used to compute the log-likelihood's Hessian by backpropagating each element of the output with respect to each element of the input. Thus, DPMs can be used for discovering leaf nodes by identifying elements in the Hessian's diagonal with a variance of zero across data samples. Applying the leaf finding algorithm d (number of nodes) times, one can discover the topological ordering of the true causal graph. At each iteration, the number of nodes decreases. We avoid re-training the score model by adjusting the score to remove the contribution of the leaf most recently removed. Our empirical results show that neural networks in this setting can be efficiently used for topological ordering in high-dimensional graphs (up to 500 nodes) and with large datasets (up to 10^5 samples).

Future Work. Denoising diffusion is an efficient way of learning the score of a distribution and computing its Jacobian. However, the method for computing the deciduous score developed in Section 3 can be used with other techniques where differentiating the score to obtain the Hessian matrix is a faster option than learning a new score, *e.g.* sliced score matching (Song et al., 2020) and approximate backpropagation (Kingma & Cun, 2010). In addition, rephrasing the causal discovery problem as topological ordering is not enough for learning the complete causal graph; spurious edges need to be pruned as a post-processing step. Therefore, future work could more comprehensively explore which feature selection methods are best suited for this task. Moreover, DPMs are generative models. Generative diffusion models in the context of causal estimation have been explored by Sanchez & Tsaftaris (2022). In this work, we have not explored the generative aspect, we have solely used DPMs as an approximation of the score.

References

Béla Bollobás, Christian Borgs, Jennifer Chayes, and Oliver Riordan. Directed Scale-Free Graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pp. 132–139, USA, 2003. Society for Industrial and Applied Mathematics.

Peter Bühlmann, Jonas Peters, and Jan Ernest. CAM: Causal Additive Models, High-dimensional Order Search And Penalized Regression. *The Annals of Statistics*, 42(6):2526–2556, 2014.

Wenyu Chen, Mathias Drton, and Y Samuel Wang. On causal discovery with an equal-variance assumption. *Biometrika*, 106(4):973–980, 12 2019.

David Maxwell Chickering. Learning Bayesian Networks is NP-Complete. In Doug Fisher and Hans-J. Lenz (eds.), *Learning from Data: Artificial Intelligence and Statistics V*, pp. 121–130. Springer New York, New York, NY, 1996.

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

Paul Erdős, Alfréd Rényi, and others. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

Nir Friedman and Daphne Koller. Being Bayesian About Network Structure. A Bayesian Approach to Structure Discovery in Bayesian Networks. *Machine Learning*, 50(1):95–125, 2003.

Asish Ghoshal and Jean Honorio. Learning linear structural equation models in polynomial time and sample complexity. In Amos Storkey and Fernando Perez-Cruz (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1466–1475. PMLR, 8 2018.

Clark Glymour, Kun Zhang, and Peter Spirtes. Review of Causal Discovery Methods Based on Graphical Models. *Frontiers in Genetics*, 10(JUN):524, 12 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances on Neural Information Processing Systems*, 2020.

Richard Zou Horace He. functorch: JAX-like composable function transforms for PyTorch. https://github.com/pytorch/functorch, 2021.

Patrik Hoyer, Dominik Janzing, Joris M Mooij, Jonas Peters, and Bernhard Schölkopf. Nonlinear causal discovery with additive noise models. In *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008.

Aapo Hyvärinen. Estimation of Non-Normalized Statistical Models by Score Matching. *Journal of Machine Learning Research*, 6:695–709, 2005.

Durk P Kingma and Yann Cun. Regularized estimation of image statistics by Score Matching. In J Lafferty, C Williams, J Shawe-Taylor, R Zemel, and A Culotta (eds.), *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient-Based Neural DAG Learning. In *International Conference on Learning Representations*, 2020.

Wai-Yin Lam, Bryan Andrews, and Joseph Ramsey. Greedy Relaxations of the Sparsest Permutation Algorithm. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

Yingzhen Li and Richard E Turner. Gradient Estimators for Implicit Models. In *International Conference on Learning Representations*, 2018.

Ignavier Ng, Shengyu Zhu, Zhuangyan Fang, Haoyang Li, Zhitang Chen, and Jun Wang. Masked Gradient-Based Causal Structure Learning. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pp. 424–432. 2022.

Young Woong Park and Diego Klabjan. Bayesian Network Learning via Topological Order. *Journal of Machine Learning Research*, 18(99):1–32, 2017.

Jonas Peters and Peter Bühlmann. Structural intervention distance for evaluating causal graphs. *Neural Computation*, 27(3):771–799, 2015.

Jonas Peters, Joris M Mooij, Dominik Janzing, and Bernhard Schölkopf. Causal Discovery with Continuous Additive Noise Models. *Journal of Machine Learning Research*, 15(58):2009–2053, 2014.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference*. MIT Press, 2017.

Garvesh Raskutti and Caroline Uhler. Learning directed acyclic graph models based on sparsest permutations. *Stat*, 7(1):e183, 1 2018.

Paul Rolland, Volkan Cevher, Matthäus Kleindessner, Chris Russell, Dominik Janzing, Bernhard Schölkopf, and Francesco Locatello. Score Matching Enables Causal Discovery of Nonlinear Additive Noise Models. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18741–18753, 2022.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data. *Science*, 308(5721): 523–529, 2005.

Pedro Sanchez and Sotirios A Tsaftaris. Diffusion Causal Models for Counterfactual Estimation. In *Proceedings of the First Conference on Causal Learning and Reasoning*, volume 177 of *Proceedings of Machine Learning Research*, pp. 647–668. PMLR, 8 2022.

Pedro Sanchez, Jeremy P Voisey, Tian Xia, Hannah I Watson, Alison Q O'Neil, and Sotirios A Tsaftaris. Causal machine learning for healthcare and precision medicine. *Royal Society Open Science*, 9(8), 8 2022.

Shohei Shimizu, Patrik O Hoyer, Aapo Hyvrinen, and Antti Kerminen. A Linear Non-Gaussian Acyclic Model for Causal Discovery. *Journal of Machine Learning Research*, 7(72):2003–2030, 2006.

Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. *Proc. of 32nd International Conference on Machine Learning*, 3:2246–2255, 12 2015.

L Solus, Y Wang, and C Uhler. Consistency guarantees for greedy permutation-based causal inference algorithms. *Biometrika*, 108(4):795–814, 8 2021.

Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced Score Matching: A Scalable Approach to Density and Score Estimation. In Ryan P Adams and Vibhav Gogate (eds.), *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pp. 574–584. PMLR, 9 2020.

Yang Song Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling Through Stochastic Differential Equations. In *ICLR*, 2021.

Peter Spirtes, Clark Glymour, and Richard Scheines. Causation, Prediction, and Search. 81, 1993.

Marc Teyssier and Daphne Koller. Ordering-Based Search: A Simple and Effective Algorithm for Learning Bayesian Networks. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pp. 584–590, Arlington, Virginia, USA, 2005. AUAI Press.

Tim Van den Bulcke, Koenraad Van Leemput, Bart Naudts, Piet van Remortel, Hongwu Ma, Alain Verschoren, Bart De Moor, and Kathleen Marchal. SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC Bioinformatics*, 7 (1):43, 2006.

Thomas Verma and Judea Pearl. Causal Networks: Semantics and Expressiveness. In Ross D SHACHTER, Tod S LEVITT, Laveen N KANAL, and John F LEMMER (eds.), *Uncertainty in Artificial Intelligence*, volume 9 of *Machine Intelligence and Pattern Recognition*, pp. 69–76. North-Holland, 1990.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 12 2011.

Xiaoqiang Wang, Yali Du, Shengyu Zhu, Liangjun Ke, Zhitang Chen, Jianye Hao, and Jun Wang. Ordering-Based Causal Discovery with Reinforcement Learning. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI*, pp. 3566–3573. International Joint Conferences on Artificial Intelligence Organization, 8 2021.

Yue Yu, Jie Chen, Tian Gao, and Mo Yu. DAG-GNN: DAG structure learning with graph neural networks. In *36th International Conference on Machine Learning, ICML 2019*, volume 2019-June, pp. 12395–12406. International Machine Learning Society (IMLS), 2019.

Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P Xing. DAGs with NO TEARS: Continuous Optimization for Structure Learning. In *n Advances in Neural Information Processing Systems*, 2018.

Xun Zheng, Chen Dan, Bryon Aragam, Pradeep Ravikumar, and Eric Xing. Learning Sparse Nonparametric DAGs. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3414–3425. PMLR, 9 2020.

A PROOFS

We re-write Equation 1 here for improved readability:

$$\nabla_{\mathrm{x}_j} \log p(\mathbf{x}) = \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_j - f_j\right)}{\partial \mathrm{x}_j} - \sum_{i \in Ch(\mathrm{x}_j)} \frac{\partial f_i}{\partial \mathrm{x}_j} \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_i - f_i\right)}{\partial x}. \tag{10}$$

A.1 PROOF LEMMA 1

Proof. We start by showing the "$\Leftarrow$" direction by deriving Equation 1 w.r.t. x_j. If x_j is a leaf, only the first term of the equation is present, then taking its derivative results in

$$\begin{aligned} \boldsymbol{H}_{l,l}(\log p(\mathbf{x})) &= \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_l - f_l\right)}{\partial x^2} \cdot \frac{df_l}{d\mathrm{x}_l} \\ &= \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_l - f_l\right)}{\partial x^2}. \end{aligned} \tag{11}$$

Therefore, only if j is a leaf and $\frac{d \log p^{\epsilon}}{dx^2} = a$, $\mathrm{Var}_{\boldsymbol{X}}\left[\boldsymbol{H}_{l,l}(\log p(\mathbf{x}))\right] = 0$. The remaining of the proof follows Rolland et al. (2022) (which was done for a Gaussian noise only), we prove by contradiction that $\Rightarrow$ is also true. In particular, if we consider that x_j is **not** a leaf and $\nabla_{\mathrm{x}_j} \log p(\mathbf{x}) = c$, with c being a constant, we can write

$$\nabla_{\mathrm{x}_j} \log p(\mathbf{x}) = c\mathrm{x}_j + g(\mathbf{x}_{-j}). \tag{12}$$

Replacing Equation 12 in to Equation 1, we have

$$c\mathrm{x}_j + g(\mathbf{x}_{-j}) = \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_j - f_j\right)}{\partial x} - \sum_{\mathrm{x}_i \in Ch(\mathrm{x}_j)} \frac{\partial f_i}{\partial \mathrm{x}_j} \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_i - f_i\right)}{\partial x}. \tag{13}$$

Let $\mathrm{x}_c \in Ch(\mathrm{x}_j)$ such that $\mathrm{x}_c \notin Pa(Ch(\mathrm{x}_j))$. x_c always exist since x_j is not a leaf, and it suffices to pick a child of x_c appearing at last position in some topological order. If we isolate the terms depending on x_c on the RHS of Equation 13, we have

$$c\mathrm{x}_j + \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_j - f_j\right)}{\partial x} - \sum_{\mathrm{x}_i \in Ch(\mathrm{x}_j), \mathrm{x}_i \neq \mathrm{x}_c} \frac{\partial f_i}{\partial \mathrm{x}_j} \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_i - f_i\right)}{\partial x} = \frac{\partial f_c}{\partial \mathrm{x}_j} \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_c - f_c\right)}{\partial x} - g(\mathbf{x}_{-j}). \tag{14}$$

Deriving both sides w.r.t. x_c, since the LHS of Equation 14 does not depend on x_c, we can write

$$\begin{aligned} &\frac{\partial}{\partial \mathrm{x}_c}\left(\frac{\partial f_c}{\partial \mathrm{x}_j} \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_c - f_c\right)}{\partial x} - g(\mathbf{x}_{-j})\right) = 0 \\ &\Rightarrow \cancelto{a}{\frac{\partial f_c}{\partial \mathrm{x}_j} \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_c - f_c\right)}{\partial x^2}} = \frac{\partial g(\mathbf{x}_{-j})}{\partial \mathrm{x}_c} \end{aligned} \tag{15}$$

Since g does not depend on x_j, $\frac{\partial f_c}{\partial \mathrm{x}_j}$ does not depend on x_j neither, implying that f_c is linear in x_j, contradicting the non-linearity assumption. □

A.2 PROOF THEOREM 1

Proof. Using Equation 1, we will derive expressions for each of the elements in Equation 6 and show that it is equivalent to δ_l in Equation 5. First, note that the score of a leaf node x_l can be denoted as:

$$\begin{aligned} \nabla_{\mathrm{x}_l} \log p(\mathbf{x}) &= \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_j - f_j\right)}{\partial \mathrm{x}_j} \\ &= \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_l - f_l\right)}{\partial x} \cdot \frac{d\left(\mathrm{x}_l - f_l\right)}{d\mathrm{x}_l} \\ &= \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_l - f_l\right)}{\partial x} \end{aligned} \tag{16}$$

Second, replacing Equation 16 into each element of $\boldsymbol{H}_l(\log p(\mathbf{x})) \in R^d$, we can write

$$\begin{aligned}\boldsymbol{H}_{l,j}(\log p(\mathbf{x})) &= \frac{\partial}{\partial \mathrm{x}_j}\left[\nabla_{\mathrm{x}_l} \log p(\mathbf{x})\right] \\ &= \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_l - f_l\right)}{\partial x^2} \cdot \frac{d\left(\mathrm{x}_l - f_l\right)}{d\mathrm{x}_j} \\ &= \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_l - f_l\right)}{\partial x^2} \cdot \frac{df_l}{d\mathrm{x}_j}.\end{aligned} \tag{17}$$

If $j = l$ in Equation 17, we have

$$\begin{aligned}\boldsymbol{H}_{l,l}(\log p(\mathbf{x})) &= \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_l - f_l\right)}{\partial x^2} \cdot \frac{df_l}{d\mathrm{x}_l} \\ &= \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_l - f_l\right)}{\partial x^2}.\end{aligned} \tag{18}$$

Finally, replacing Equations 16, 17 and 11 into the Equation 5 for a single node x_j, if $j \neq l$, we have

$$\begin{aligned}\delta_j &= \frac{\boldsymbol{H}_{l,j}(\log p(\mathbf{x})) \cdot \nabla_{\mathrm{x}_l} \log p(\mathbf{x})}{\boldsymbol{H}_{l,l}(\log p(\mathbf{x}))} \\ &= \frac{\frac{\partial \log p^{\epsilon}(\mathrm{x}_l - f_l)}{\partial x^2} \cdot \frac{df_l}{d\mathrm{x}_j} \cdot \frac{\partial \log p^{\epsilon}(\mathrm{x}_l - f_l)}{\partial x}}{\frac{\partial \log p^{\epsilon}(\mathrm{x}_l - f_l)}{\partial x^2}} \\ &= \frac{df_l}{d\mathrm{x}_j} \cdot \frac{\partial \log p^{\epsilon}\left(\mathrm{x}_l - f_l\right)}{\partial x}.\end{aligned} \tag{19}$$

The last line in Equation 19 is the same as in Equation 5 from Lemma 2, proving that δ_j can be written using the first and second order derivative of the log-likelihood. □

B Score of Nonlinear ANM with Gaussian noise

A SCM entails a distribution

$$p(\mathbf{x}) = \prod_{i=1}^{d} p(\mathrm{x}_i \mid Pa(\mathrm{x}_i)), \tag{20}$$

over the variables $\mathbf{x}$ (Peters et al., 2017) By assuming that the noise variables $\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$ and inserting the ANM function, Equation 20 can be written as

$$\log p(\mathbf{x}) = -\frac{1}{2}\sum_{i=1}^{d}\left(\frac{\mathrm{x}_i - f_i(Pa(\mathrm{x}_i))}{\sigma_i}\right)^2 - \frac{1}{2}\sum_{i=1}^{d}\log(2\pi\sigma_i^2). \tag{21}$$

The score of $p(\mathbf{x})$ can hence be written as

$$\nabla_{\mathrm{x}_j} \log p(\mathbf{x}) = -\frac{\mathrm{x}_j - f_j(Pa(\mathrm{x}_j))}{\sigma_j^2} + \sum_{i \in \text{children}(j)} \frac{\partial f_i}{\partial \mathrm{x}_j}(Pa(\mathrm{x}_i))\frac{\mathrm{x}_i - f_i(Pa(\mathrm{x}_i))}{\sigma_i^2}. \tag{22}$$

C Visualisation of the Score's Jacobian for Two Variables

Considering a two variables problem where the causal mechanisms are $B = f_\omega(A) + \epsilon_B$ and $A = \epsilon_A$ with $\epsilon_A, \epsilon_B \sim \mathcal{N}(0, 1)$ and f_ω being a two-layer MLP with randomly initialised weights. Note that the variance of $\frac{\partial^2 \log p(\boldsymbol{A},\boldsymbol{B})}{\partial \boldsymbol{B}^2}$, while not 0 as predicted by Equation 2, is smaller than $\frac{\partial^2 \log p(\boldsymbol{A},\boldsymbol{B})}{\partial \boldsymbol{A}^2}$ allowing discovery of the true causal direction.

D Experiments Details

D.1 Hyperparameters of DPM training

We now describe the hyperparameters for the diffusion training. We use number of time steps $T = 100$, β_t is a linearly scheduled between $\beta_{\text{min}} = 0.0001$ and $\beta_{\text{max}} = 0.02$. The model is trained

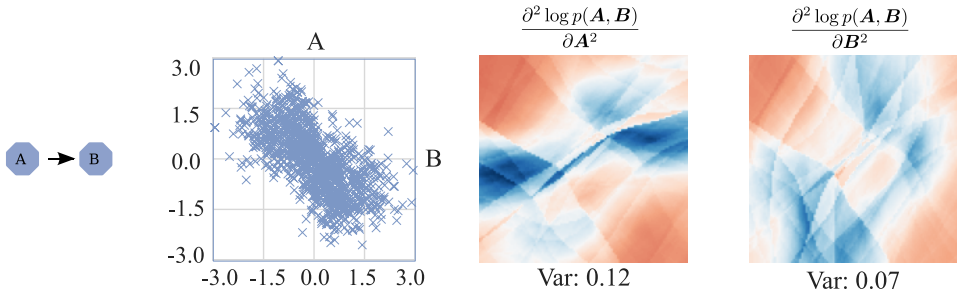

Figure 5: Visualisation of the diagonal of the score's Jacobian estimated with a diffusion model as in Equation 7 for a two-variable SCM where $A \to B$.

Table 2: MLP architecture. The hyperparameters of each Linear layer depend on d such that big $= max(1024, 5 * d)$ and small $= max(128, 3 * d)$.

Layer	**Hyperparameters**
Linear	Ch: $(d + 1$, small)
LeakyReLU	
LayerNorm	
Dropout	Prob: 0.2
Linear	Ch: (small, big)
LeakyReLU	
LayerNorm	
Linear	Ch: (big, big)
LeakyReLU	
Linear	Ch: (big, big)
LeakyReLU	
Linear	Ch: (big, d)

according to Equation 3 which follows Ho et al. (2020). During sampling, t is sampled from a Uniform distribution.

D.2 Neural Architecture

The neural network follows a simple MLP with 5 Linear layers, LeakyReLU activation function, Layer Normalization and Dropout in the first layer. The full architecture is detailed in Table 2.

D.3 Metrics

For each method, we compute the

SHD. Structural Hamming distance between the output and the true causal graph, which counts the number of missing, falsely detected, or reversed edges.

SID. Structural Intervention Distance is based on a graphical criterion only and quantifies the closeness between two DAGs in terms of their corresponding causal inference statements(Peters & Bühlmann, 2015).

Order Divergence. Rolland et al. (2022) propose this quantity for measuring how well the topological order is estimated. For an ordering π, and a target adjacency matrix A, we define the topological order divergence $D_{top}(\pi, A)$ as

$$D_{top}(\pi, \boldsymbol{A}) = \sum_{i=1}^{d} \sum_{j:\pi_i > \pi_j} \boldsymbol{A}_{ij}. \tag{23}$$

If π is a correct topological order for $\boldsymbol{A}$, then $D_{top}(\pi, \boldsymbol{A}) = 0$. Otherwise, $D_{top}(\pi, \boldsymbol{A})$ counts the number of edges that cannot be recovered due to the choice of topological order. Therefore, it provides a lower bound on the SHD of the final algorithm (irrespective of the pruning method).

E OTHER RESULTS

E.1 OVERFITTING

The data used for topological ordering (inference) is a subset of the training data. Therefore, it is not obvious if overfitting would be an issue with our algorithm. Therefore, we run an experiment where we fix the number of epochs to 2000 considered high for a set of runs and use early stopping for another set in order to verify if overfitting is an issue. On average across all 20 nodes datasets, the early stopping strategy output an ordering diverge of 9.5 whilst overfitting is at 11.1 showing that the method does not benefit from overfitting.

E.2 OPTIMAL t FOR SCORE ESTIMATION

As noted by Vincent (2011), the best approximation of the score by a learned denoising function is when the training signal-to-noise (SNR) ratio is low. In diffusion model training, $t = 0$ corresponds to the coefficient with lowest SNR. However, we found empirically that the best score estimate varies somehow randomly across different values of t. Therefore, we run the the leaf finding function (Equation 9) N times for values of t evenly spaced in the $[0, T]$ interval and choose the best leaf based on majority vote. We show in Figure 6 that majority voting is a better approach than choosing a constant value for t.

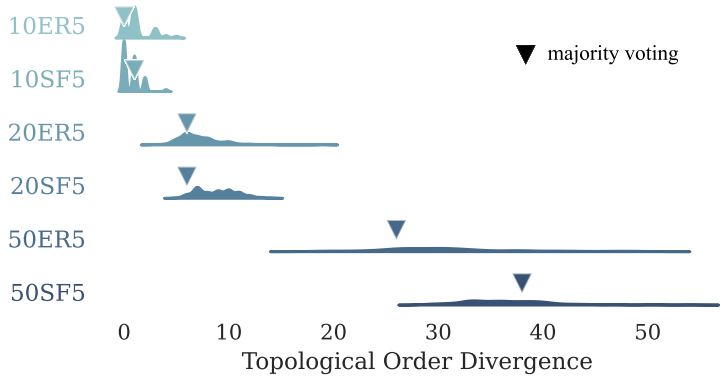

Figure 6: The distribution of order divergence measured for different values of t is highly variable. Therefore, we show that we obtain a better approximation with majority voting.

E.3 ABLATIONS OF DIFFAN MASKING AND GREEDY

We now verify how DiffAN masking and DiffAN greedy compare against the original version detailed in the main text which computes the deciduous score. Here, we use the same datasets decribed in Section 5.1 which comprises 4 (20ER1, 20ER5, 20SF1, 20SF5) synthetic dataset types with 27 variations over seeds, noise type and noise scale.

We observe, in Figure 7, that the greedy version performs the best but it is the slowest, as seen in Figure 8. DiffAN masking

E.4 DETAILED RESULTS

We present the numerical results for the violinplots in Section 5.1 in Tables 3 and 4. The results are presented in mean_{std} with statistics acquired over experiments with 3 seeds.

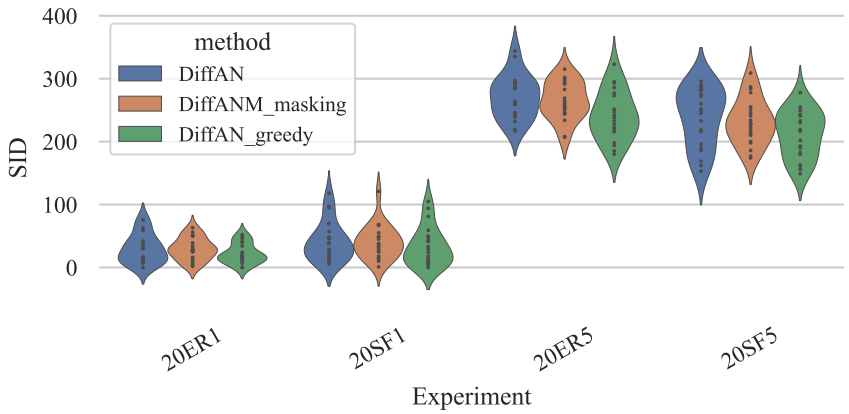

Figure 7: SID metric for different versions of DiffAN.

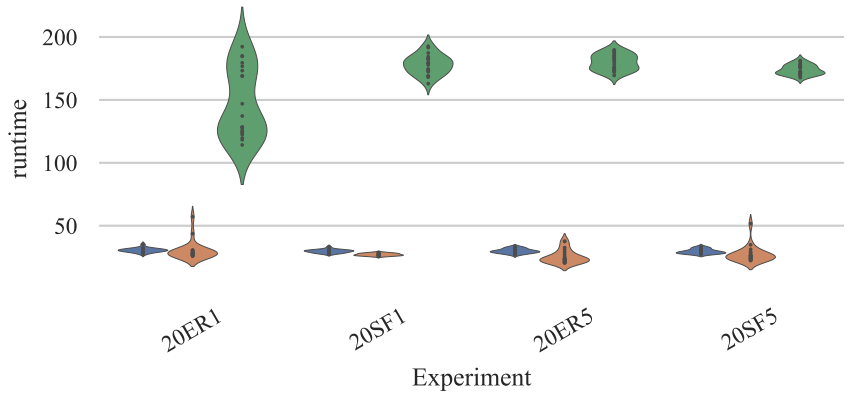

Figure 8: Runtime in seconds for different versions of DiffAN.

Table 3: Erdös-Rényi (ER) graphs.

Exp Name	noisetype	method	shd	sid	runtime (seconds)
20ER1	exp	CAM	$6.50_{3.21}$	$35.50_{27.54}$	$112.17_{69.57}$
		DiffAN	$7.67_{4.68}$	$38.83_{24.94}$	$30.07_{1.60}$
		DiffAN Greedy	$6.67_{4.68}$	$34.17_{18.37}$	$121.66_{4.19}$
		GranDAG	$12.83_{5.60}$	$65.17_{21.99}$	$341.69_{74.82}$
		SCORE	$4.17_{2.86}$	$17.50_{11.26}$	$50.49_{41.87}$
	gauss	CAM	$1.33_{1.03}$	$6.50_{6.69}$	$83.27_{11.75}$
		DiffAN	$8.17_{4.02}$	$41.67_{24.21}$	$30.98_{2.28}$
		DiffAN Greedy	$3.33_{2.16}$	$18.33_{11.40}$	$129.10_{9.54}$
		GranDAG	$10.50_{8.62}$	$44.50_{37.90}$	$304.07_{17.39}$
		SCORE	$8.67_{4.13}$	$41.67_{21.54}$	$24.77_{0.20}$
	laplace	CAM	$0.83_{0.98}$	$4.00_{6.96}$	$90.85_{15.49}$
		DiffAN	$2.33_{2.07}$	$8.67_{4.68}$	$31.01_{1.18}$
		DiffAN Greedy	$3.00_{1.67}$	$14.33_{4.76}$	$175.30_{19.71}$
		GranDAG	$10.33_{5.82}$	$41.17_{24.98}$	$302.00_{16.23}$
		SCORE	$4.17_{3.43}$	$24.17_{19.57}$	$26.62_{1.88}$
20ER5	exp	CAM	$60.67_{9.77}$	$240.00_{33.02}$	$105.65_{27.49}$
		DiffAN	$67.50_{4.32}$	$278.83_{22.83}$	$30.33_{2.12}$
		DiffAN Greedy	$63.00_{6.07}$	$266.33_{50.25}$	$186.21_{3.28}$
		GranDAG	$80.17_{7.99}$	$304.17_{26.90}$	$310.24_{13.71}$
		SCORE	$55.83_{7.55}$	$190.67_{32.91}$	$18.41_{3.73}$
	gauss	CAM	$64.67_{5.85}$	$214.67_{11.11}$	$130.82_{26.10}$
		DiffAN	$68.50_{8.12}$	$289.00_{42.31}$	$29.58_{1.99}$
		DiffAN Greedy	$63.17_{6.18}$	$228.83_{32.86}$	$178.17_{7.17}$
		GranDAG	$81.00_{6.72}$	$273.00_{46.72}$	$319.17_{18.81}$
		SCORE	$63.50_{6.69}$	$233.83_{39.65}$	$21.29_{6.63}$
	laplace	CAM	$68.00_{7.04}$	$228.50_{27.86}$	$157.70_{22.55}$
		DiffAN	$68.83_{7.86}$	$248.67_{30.36}$	$29.52_{1.73}$
		DiffAN Greedy	$67.33_{8.55}$	$238.00_{35.19}$	$176.71_{4.80}$
		GranDAG	$82.67_{6.47}$	$271.33_{35.19}$	$305.29_{13.93}$
		SCORE	$66.33_{7.09}$	$218.67_{25.81}$	$23.01_{4.33}$

Table 4: Scale Free (SF) graphs.

Exp Name	noisetype	method	shd	sid	runtime (seconds)
20SF1	exp	CAM	$7.17_{2.86}$	$32.67_{19.54}$	$85.08_{27.71}$
		DiffAN	$8.67_{2.80}$	$38.67_{20.99}$	$29.23_{1.36}$
		DiffAN Greedy	$9.17_{2.32}$	$35.33_{17.26}$	$183.80_{4.81}$
		GranDAG	$15.50_{2.59}$	$68.67_{24.19}$	$298.88_{4.96}$
		SCORE	$5.83_{5.04}$	$23.50_{23.65}$	$26.13_{1.39}$
	gauss	CAM	$1.83_{2.23}$	$9.17_{10.21}$	$84.46_{19.07}$
		DiffAN	$9.83_{6.24}$	$56.67_{41.59}$	$30.19_{0.92}$
		DiffAN Greedy	$7.33_{5.32}$	$46.83_{38.42}$	$175.63_{8.82}$
		GranDAG	$12.17_{2.71}$	$46.67_{16.22}$	$299.25_{13.14}$
		SCORE	$8.17_{4.45}$	$41.67_{28.75}$	$26.29_{1.83}$
	laplace	CAM	$2.83_{3.92}$	$7.83_{10.93}$	$85.15_{21.84}$
		DiffAN	$6.00_{3.74}$	$24.17_{16.18}$	$29.68_{2.00}$
		DiffAN Greedy	$4.83_{3.66}$	$17.17_{17.47}$	$177.05_{8.82}$
		GranDAG	$9.50_{3.27}$	$35.67_{20.84}$	$295.12_{2.88}$
		SCORE	$5.67_{3.50}$	$22.50_{15.27}$	$26.39_{1.79}$
20SF5	exp	CAM	$47.83_{5.78}$	$228.83_{53.89}$	$113.82_{13.32}$
		DiffAN	$46.83_{11.48}$	$243.50_{34.68}$	$30.03_{2.17}$
		DiffAN Greedy	$43.50_{6.89}$	$236.83_{23.17}$	$173.13_{3.52}$
		GranDAG	$60.67_{8.80}$	$275.00_{22.56}$	$284.34_{5.33}$
		SCORE	$38.50_{9.14}$	$180.33_{57.44}$	$19.79_{2.92}$
	gauss	CAM	$46.83_{8.11}$	$199.17_{53.13}$	$130.41_{20.55}$
		DiffAN	$50.83_{4.62}$	$259.50_{47.54}$	$28.58_{1.16}$
		DiffAN Greedy	$45.17_{6.15}$	$224.17_{41.80}$	$174.90_{3.69}$
		GranDAG	$61.67_{4.03}$	$241.67_{43.11}$	$292.77_{7.29}$
		SCORE	$44.50_{5.24}$	$217.83_{50.05}$	$19.77_{2.51}$
	laplace	CAM	$49.50_{9.01}$	$191.33_{27.43}$	$160.53_{23.16}$
		DiffAN	$52.00_{5.66}$	$230.67_{50.66}$	$29.05_{1.60}$
		DiffAN Greedy	$47.00_{9.94}$	$191.33_{26.16}$	$173.42_{3.52}$
		GranDAG	$65.00_{7.13}$	$262.50_{44.23}$	$295.40_{7.72}$
		SCORE	$46.67_{10.03}$	$193.83_{34.14}$	$43.84_{27.14}$