paper framework: 🟨
Framework in detail: 〰
Implementation detail: 🟩

theory to learn: 🟪 〰
Helpful for my research. 🟦
Confused points: 🟥 〰

Efficient Neural Causal Discovery without Acyclicity Constraints

Phillip Lippe
University of Amsterdam
QUVA lab
p.lippe@uva.nl

Taco Cohen
Qualcomm AI Research*
tacos@qti.qualcomm.com

Efstratios Gavves
University of Amsterdam
QUVA lab
e.gavves@uva.nl

Abstract

Learning the structure of a causal graphical model using both observational and interventional data is a fundamental problem in many scientific fields. A promising direction is continuous optimization for score-based methods, which efficiently learn the causal graph in a data-driven manner. However, to date, those methods require constrained optimization to enforce acyclicity or lack convergence guarantees. In this paper, we present ENCO, an efficient structure learning method for directed, acyclic causal graphs leveraging observational and interventional data. ENCO formulates the graph search as an optimization of independent edge likelihoods with the edge orientation being modeled as a separate parameter. Consequently, we can provide convergence guarantees of ENCO under mild conditions without constraining the score function with respect to acyclicity. In experiments, we show that ENCO can efficiently recover graphs with hundreds of nodes, an order of magnitude larger than what was previously possible, while handling deterministic variables and latent confounders.

1 Introduction

Uncovering and understanding causal mechanisms is an important problem not only in machine learning [3, 31, 39] but also in various scientific disciplines such as computational biology [10, 28, 37], epidemiology [36, 46], and economics [15, 31]. A common task of interest is *causal structure learning* [31, 33] which aims at learning a directed acyclic graph (DAG) in which edges represent causal relations between variables. While observational data alone is in general not sufficient to identify the DAG [13, 49], interventional data can improve identifiability up to finding the exact graph [8, 9]. With recent advances in gene editing technologies providing large amounts of interventional gene expression data [7, 24], there is a need for algorithms that can perform structure learning for graphs with several hundreds of nodes [7, 24].

Finding the right DAG is challenging as the solution space grows super-exponentially with the number of variables. A promising new direction are continuous-optimization methods [3, 4, 18, 51, 52, 53, 54] that are more computationally efficient than previous score-based and constraint-based methods [12, 33] while leveraging the expressiveness of neural networks as function approximators. To restrict the search space to acyclic graphs, Zheng et al. [52] first proposed to view the search as a constrained optimization problem using an augmented Lagrangian procedure to solve it. Several follow-up works improved the process [4, 51, 53]. An alternative approach is the usage of a regularizer in the learning objective that penalizes cyclic graphs and is simpler to optimize [18, 54]. Nonetheless, methods with such regularizers commonly lack guarantees for converging to the correct causal graph. To date, continuous optimization methods do not scale well to more than 50 variables due to a discrete search space and difficulties in enforcing acyclicity.

why?

[*] Qualcomm AI Research in an initiative of Qualcomm Technologies, Inc.

Preprint. Under review.

How to generate
interventional data

In this work, we show that with suitable interventional data, we do not need to limit the search space to DAGs in the first place. Instead, by modeling the orientation of an edge as a separate parameter, the optimization of a likelihood-based objective already converges to the correct, acyclic graph. The parameterization of edge orientations allows us to derive low-variance gradient estimators for the discrete search space and give convergence guarantees under mild conditions. We call this method ENCO, **E**fficient **N**eural **C**ausal Disc**o**very. In experiments, we show that ENCO handles various graph settings well, and even recovers graphs with up to 1,000 nodes in less than nine hours of compute using a single GPU (NVIDIA RTX3090) while having less than one mistake on average out of 1 million possible edges. Further, we show how ENCO can handle and detect latent confounders.

Our contributions are summarized as follows. Firstly, we propose ENCO, a causal structure learning method for observational and interventional data using continuous optimization. ENCO models the edge orientation as separate parameter removing the need of acyclicity constraints. Secondly, we derive unbiased, low-variance gradient estimators and show that with sufficient data and under mild assumptions, ENCO converges to the underlying, correct causal graph. Finally, we provide an extensive comparison of ENCO to state-of-the-art methods including real-world datasets.

2 Background and Related Work

2.1 Causal graphical models

A causal graphical model (CGM) is defined by a distribution $\mathbb{P}$ over a set of random variables $\boldsymbol{X} = \{X_1, ..., X_N\}$ and a directed acyclic graph (DAG) $G = (V, E)$. In this graph, each node $i \in V$ corresponds to the random variable X_i and an edge $(i, j) \in E$ represents a direct causal relation from variable X_i to X_j: $X_i \to X_j$. A DAG G and a joint distribution $\mathbb{P}$ are faithful to each other if all and only the conditional independencies present in $\mathbb{P}$ are entailed by G [30]. Vice versa, a distribution $\mathbb{P}$ is Markov to a graph G if the joint distribution can be factorized as $p(\boldsymbol{X}) = \prod_{i=1}^{N} p_i(X_i|\text{pa}(X_i))$ where $\text{pa}(X_i)$ is the set of parents of the node i in G. An important concept which distinguishes CGMs from standard Bayesian Networks are interventions. An intervention on a variable X_i describes the local replacement of its conditional distribution $p_i(X_i|\text{pa}(X_i))$ by a new distribution $\tilde{p}(X_i|\text{pa}(X_i))$. X_i is thereby referred to as the intervention target. In this work, we focus on perfect interventions for which the new distribution is independent of the original parents: $\tilde{p}(X_i|\text{pa}(X_i)) = \tilde{p}(X_i)$.

2.2 Causal structure learning

Discovering the graph G from samples of a joint distribution $\mathbb{P}$ is called *causal structure learning* or *causal discovery*, which is a fundamental problem in causality [31, 33]. Samples obtained from $\mathbb{P}$ without any interventions are called observational data. Many algorithms have been developed for learning the causal structure from observational and/or interventional data, and are commonly grouped into constraint-based and score-based.

Constraint-based methods use conditional independence tests to identify edges in the causal structure [14, 16, 27, 32, 42, 43, 44]. For instance, the Invariant Causal Prediction (ICP) algorithm [5, 32] learns the graph structure by exploiting the fact that causal mechanisms remain unchanged under an intervention except the one intervened on [31, 38]. We rely on a similar mechanism by comparing graphs based on how well their distributions fitted to observational data generalize to interventions.

Score-based methods, on the other hand, search through the space of all possible causal structures with the goal of optimizing a specified metric [11, 13, 18, 45, 54]. This metric, also referred to as score function, is usually a combination of how well the structure fits the data, for instance in terms of log-likelihood, as well as regularizers for encouraging sparsity. Since the search space of DAGs is super-exponential in the number of nodes, many score-based approaches rely on a greedy search [13, 26, 47, 49]. For instance, GIES [13] repeatedly adds, removes, and flips the directions of edges in a proposal graph until no higher-scoring graph can be found. The Interventional Greedy SP (IGSP) algorithm [47] is a hybrid method using conditional independence tests in its score function.

Continuous optimization methods are similar to score-based methods but avoid the combinatorial greedy search over DAGs by using gradient-based methods [3, 4, 18, 51, 52, 53, 54]. Thereby, the adjacency matrix is parameterized by weights that represent linear factors or probabilities of having an edge between a pair of nodes. The main challenge of such methods is how to limit the search space

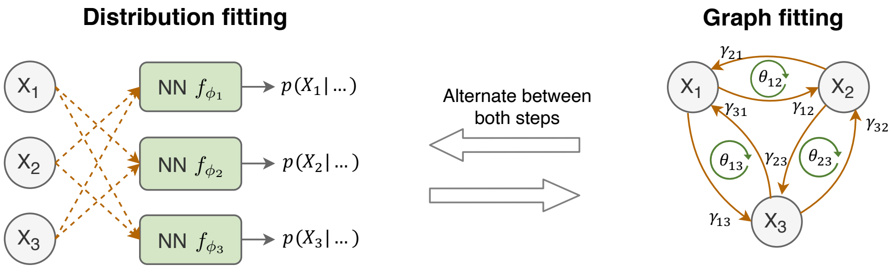


Figure 1: Visualization of the two training stages of ENCO, distribution fitting and graph fitting, on an example graph with 3 variables (X_1, X_2, X_3). The graph on right further shows how the parameters γ and $\boldsymbol{\theta}$ correspond to edge probabilities. We learn those parameters by comparing multiple graph samples on how well they generalize from observational to interventional data.

to acyclic graphs. One common approach is to view the search as a constrained optimization problem and deploy an augmented Lagrangian procedure to solve it [4, 51, 52, 53]. Methods that deploy such a constraint include NOTEARS [52] and DCDI [4]. As an alternative to a constrained optimization procedure, Ke et al. [18] propose to use a regularization term that penalizes cyclic graphs. This allows for simpler optimization as standard gradient-based methods can be deployed without constraints. However, the regularizer needs to be designed such that the correct, acyclic causal graph is the global optimum of the score function. In this work, we show that when having perfect interventions for each variable separately, we can define the score function without any acyclicity constraints or regularizers. This also allows us to scale up our approach to graphs with more than 1,000 nodes without a significant drop in accuracy, and provide a convergence guarantee under mild conditions.

3 Efficient Neural Causal Discovery

We first discuss the context in which we aim at performing ① causal structure learning with ENCO. We then detail the ② method and its training procedure in Sections 3.2 and 3.3. Finally, we discuss ③ convergence guarantees of the method and its extension to latent confounders.

3.1 Scope and Assumptions

Notes: causal assumptions {
- Causal sufficiency assumption.
- Causal Markov assumption.
- Causal Faithfulness assumption.

We consider the task of finding a directed acyclic graph $G = (V, E)$ of an unknown CGM given observational and interventional samples. We start by assuming that the CGM is causally sufficient, ① *i.e.*, all common causes of variables are included in the DAG and observable. In the following, however, we also discuss extensions for inferring the causal mechanisms in graphs with latent ② confounding causal variables. We emphasize that we place no constraints on the domains of the variables: they can be discrete, continuous, or mixed.

The scope with respect to the interventions closely follows Ke et al. [18]. The interventional data is created via sparse interventions that only affect a single variable, and are retracted before the next intervention is performed. Furthermore, we consider that interventions are perfect, that is the new variable distribution is independent of the parents. We consider that interventions have been performed for every variable and samples from it including the intervention target are provided. Last, we emphasize that we do not require specific distributions in the interventions.

TODO: (Read code in this part)

3.2 Overview

To learn a causal graph from observational and interventional data, ENCO models a probability for every possible directed edge between pairs of variables. The goal is to optimize these probabilities such that all edges in the ground truth graph converge to one, and all others to zero. The optimization exploits the idea of independent causal mechanisms [31, 32, 38]: we search for the graph which generalizes best from observational to interventional data. In the ground-truth causal graph, the conditional distributions for all variables stay invariant under an intervention except the intervened ones. Meanwhile, this does not hold for graphs that model the same distribution but with a flipped, missing or additional edge [32]. To implement this optimization, we alternate between two learning stages visualized in Figure 1: distribution fitting and graph fitting.

Distribution fitting trains a neural network f_{ϕ_i} per variable X_i to model its observational, conditional data distribution $p(X_i|...)$. The input to the network are all other variables, $\boldsymbol{X}_{-i}$, but we apply a dropout-like scheme to the input for simulating different sets of parents. Specifically, during training, we randomly set an input variable X_j to zero based on the probability of its corresponding edge $X_j \to X_i$. Similar techniques have been used by previous works [4, 17, 18, 22, 50]. The training can be summarized as the following optimization problem:

$$\min_{\phi_i} \mathbb{E}_{\boldsymbol{X}} \mathbb{E}_{\boldsymbol{M}} \left[-\log f_{\phi_i}(X_i; \boldsymbol{M}_{-i} \odot \boldsymbol{X}_{-i}) \right] \tag{1}$$

where $M_j \sim \text{Ber}(p(X_j \to X_i))$. If X_i is a categorical random variable, we apply a softmax as output activation function of f_{ϕ_i}. For continuous cases, a Normalizing Flow [35] can be used for f_{ϕ_i}.

Graph fitting uses the learned networks and interventional data to score and compare different graphs. For parameterizing the edge probabilities, we use two sets of parameters: $\boldsymbol{\gamma} \in \mathbb{R}^{N \times N}$ represents the existence of edges in a graph, and $\boldsymbol{\theta} \in \mathbb{R}^{N \times N}$ the orientation of the edges. The likelihood of an edge is determined by $p(X_i \to X_j) = \sigma(\gamma_{ij}) \cdot \sigma(\theta_{ij})$, with $\sigma(...)$ being the sigmoid function and $\theta_{ij} = -\theta_{ji}$. The probability of the two orientations always sum to one. The benefit of separating the edge probabilities into two independent parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$ is that it gives us more control over the gradient updates. The existence of an (undirected) edge (31) can usually be already learned from observational or arbitrary interventional data alone excluding deterministic variables [31]. In contrast, the orientation can only be reliably detected from data for which an intervention is performed on its adjacent nodes, *i.e.* X_i or X_j for learning θ_{ij}. While other interventions eventually provide information on the edge direction, *e.g.*, intervening on a node X_k which is a child of X_i and a parent of X_j, we do not know the relation of X_k to X_i and X_j at this stage, as we are in the process of learning the structure. Hence, only the interventions on X_i or X_j reliably uncover the orientation θ_{ij}. Despite having just one variable for the orientation, γ_{ij} and γ_{ji} are learned as two independent parameters. This is because on interventional data, an edge can improve the log-likelihood estimate in one direction, but not necessarily the other as well leading to conflicting gradients.

The objective function we use for optimizing the graph parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$ is written as:

the existence of edges

$$\tilde{\mathcal{L}} = \underbrace{\mathbb{E}_{\hat{I}\sim p_I(I)} \mathbb{E}_{\tilde{p}_{\hat{I}}(\boldsymbol{X})}}_{\text{interventional data distribution.}} \overset{\text{adj matrix}}{\boxed{\mathbb{E}_{p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C)}}} \left[\sum_{i=1}^{N} \mathcal{L}_C(X_i) \right] + \lambda_{\text{sparse}} \sum_{i=1}^{N} \sum_{j=1}^{N} \sigma(\gamma_{ij}) \cdot \sigma(\theta_{ij}) \tag{2}$$

update gradient.

where $p_I(I)$ is the distribution over which variable to intervene on (usually uniform), and $\tilde{p}_{\hat{I}}(\boldsymbol{X})$ the joint distribution of all variables under the intervention $\hat{I}$. In other words, these two distributions represent our interventional data distribution. The distribution over adjacency matrices C under $\boldsymbol{\gamma}, \boldsymbol{\theta}$ is denoted by $p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C)$ with $C_{ij} \sim \text{Ber}(\sigma(\gamma_{ij})\sigma(\theta_{ij}))$, and $\mathcal{L}_C(X_i)$ is the negative log-likelihood estimate of variable X_i conditioned on the parents according to C: $\mathcal{L}_C(X_i) = -\log f_{\phi_i}(X_i; C_{\cdot,i} \odot \boldsymbol{X}_{-i})$. The second term of Equation 2 represents a prior towards sparser graphs removing redundant edges. It is an ℓ_1-regularizer on the edge probabilities with the hyperparameter λ_{sparse} as regularization weight. The goal is to optimize $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$ such that it minimizes the objective $\tilde{\mathcal{L}}$.

Prediction Alternating between distribution and graph fitting stage allows us to fine-tune the neural networks to the most probable parent sets along the training. After training, we obtain a graph prediction by selecting the edges for which $\sigma(\gamma_{ij})$ and $\sigma(\theta_{ij})$ are greater than 0.5. The orientation parameters prevent loops between any two variables, since $\sigma(\theta_{ij})$ can only be greater than 0.5 in one direction. Although the orientation parameters do not guarantee the absence of loops with more variables at any stage of the training, we show that ENCO converges to the correct, acyclic graph.

understand details in this part.

3.3 Low-variance gradient estimators for edge parameters

To update $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$ based on the objective in Equation 2, we need to determine their gradients through the expectation $\mathbb{E}_{p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C)}$ where C is a discrete variable. For this, we apply REINFORCE [48]. For clarity of exposition, we limit the discussion here to the final results and provide the detailed derivations in Appendix A. For the parameter γ_{ij}, we obtain the following gradient:

$$\frac{\partial}{\partial \gamma_{ij}} \tilde{\mathcal{L}} = \sigma'(\gamma_{ij}) \cdot \sigma(\theta_{ij}) \cdot \mathbb{E}_{\boldsymbol{X}, C_{-ij}} \left[\mathcal{L}_{X_i \to X_j}(X_j) - \mathcal{L}_{X_i \not\to X_j}(X_j) + \lambda_{\text{sparse}} \right] \tag{3}$$

where $\mathbb{E}_{\boldsymbol{X},C_{-ij}}$ summarizes for brevity the three expectations in Equation 2 up to the edge $X_i \rightarrow X_j$. This excludes interventions on X_j since we assume the interventions to be perfect. Further, $\mathcal{L}_{X_i \rightarrow X_j}(X_j)$ denotes the negative log likelihood for X_j under the adjacency matrix C_{-ij} including an edge from X_i to X_j, *i.e.* $C_{ij} = 1$. The gradient in Equation 3 can be intuitively explained: if by the addition of the edge $X_i \rightarrow X_j$, the log-likelihood estimate of X_j is improved by more than λ_{sparse}, we increase the corresponding edge parameter γ_{ij}; otherwise, we decrease it.

The orientation parameters $\boldsymbol{\theta}$ are similarly derived as $\boldsymbol{\gamma}$. As mentioned before, we only want to take the gradients for θ_{ij} when the intervention is performed on X_i or X_j. This leads us to:

$$\frac{\partial}{\partial \theta_{ij}} \tilde{\mathcal{L}} = \sigma'(\theta_{ij}) \Big(p(I_{X_i}) \cdot \sigma(\gamma_{ij}) \cdot \mathbb{E}_{I_{X_i},\boldsymbol{X},C_{-ij}} \left[\mathcal{L}_{X_i \rightarrow X_j}(X_j) - \mathcal{L}_{X_i \not\rightarrow X_j}(X_j) \right] - \\ p(I_{X_j}) \cdot \sigma(\gamma_{ji}) \cdot \mathbb{E}_{I_{X_j},\boldsymbol{X},C_{-ij}} \left[\mathcal{L}_{X_j \rightarrow X_i}(X_i) - \mathcal{L}_{X_j \not\rightarrow X_i}(X_i) \right] \Big) \tag{4}$$

The probability of taking an intervention on X_i is represented by $p(I_{X_i})$ (usually uniform across variables). When the edge $X_i \rightarrow X_j$ improves the log-likelihood of X_j under intervention on X_i, then the gradient increases θ_{ij}. In contrast, when the true edge is $X_j \rightarrow X_i$, the correlation between X_i and X_j learned from observational data would yield a worse likelihood estimate on interventional data than without the edge $X_j \rightarrow X_i$. This is because $p(X_j|X_i, ...)$ does not stay invariant under intervening on X_i. Lastly, for independent nodes, the expectation of the gradient is zero.

Based on Equations 3 and 4, we obtain a tractable, unbiased gradient estimator by using Monte-Carlo sampling. Luckily, samples can be shared across variables making training efficient. We first sample an intervention, a corresponding data batch, and K graphs from $p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C)$ (usually between 20 and 100 for K). We then evaluate the log likelihoods of all variables for these graphs on the batch. Those results are used to estimate $\mathcal{L}_{X_i \rightarrow X_j}(X_j)$ and $\mathcal{L}_{X_i \not\rightarrow X_j}(X_j)$ for all pairs of variables X_i and X_j by simply averaging the results for the two cases separately. Finally, the likelihood estimates are used to determine the gradients for $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$.

Previous methods [3, 4, 18] relied on a different REINFORCE-like estimator proposed by Bengio et al. [3]. For the parameter γ_{ij}, for instance, the gradient under this estimator looks as follows:

$$g_{ij} = \sigma(\theta_{ij}) \cdot \mathbb{E}_{\boldsymbol{X}} \left[\frac{\mathbb{E}_C \left[(\sigma(\gamma_{ij}) - C_{ij}) \cdot \mathcal{L}_C(X_j) \right]}{\mathbb{E}_C \left[\mathcal{L}_C(X_j) \right]} + \lambda_{\text{sparse}} \right] \tag{5}$$

where g_{ij} represents the gradient of γ_{ij}. Performing Monte-Carlo sampling for estimating the gradient leads to a biased estimate which becomes asymptotically unbiased with increasing number of samples [3]. The division by the expectation of $\mathcal{L}_C(X_j)$ is done for variance reduction [25]. A major benefit of our gradient formulation in Equation 3 is that is provides a 10 times lower standard deviation as visualized in Figure 2. This is because Equation 5 is sensitive to the proportion of C_{ij} being one or zero, while Equation 3 removes this noise by considering the difference of the two independent Monte-Carlo estimates $\mathcal{L}_{X_i \rightarrow X_j}(X_j)$ and $\mathcal{L}_{X_i \not\rightarrow X_j}(X_j)$. Hence, we can use a smaller sample size than previous methods making ENCO much more efficient.

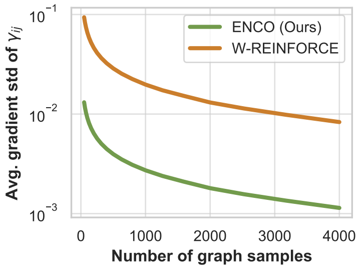


Figure 2: Comparing the variance of ENCO's gradient estimator to previous REINFORCE-like estimators [3, 18] over number of graph samples.

The objective in Equation 2 does not exclude cyclic graphs and might be minimized by one. To prevent this, other score-based methods commonly add a DAG regularizer penalizing cyclic graphs [18, 54] or use constraint-based optimization [4, 52, 53]. In ENCO, the orientation parameters $\boldsymbol{\theta}$ already prevent bidirectional edges between any two variables. We will show next that $\boldsymbol{\theta}$ converges to the correct orientation for every ancestor-descendant pair simply based on the gradients of the objective in Equation 2.

3.4 Convergence guarantees

Next, we need to analyze under which assumptions we can guarantee that ENCO convergences to the correct causal graph given sufficient time and data. As we perform gradient-based optimization, this is equivalent to showing when the objective of Equation 2 does not contain any local minima. The proof for these guarantees is detailed in Appendix B, and we limit our discussion here to the main assumptions and implications. Throughout the section, we assume that the neural networks trained in distribution fitting accurately model the true conditional probabilities $p(X_i|...)$.

We structure this proof in three steps and show under which conditions *(i)* the orientation parameters $\boldsymbol{\theta}$ converge to the correct values for every ancestor-descendant pair, *(ii)* the edge existence parameters $\boldsymbol{\gamma}$ converge for edges in the ground truth graph, and *(iii)* all other edges converge to a probability of zero. First, we discuss the orientation parameters:

Lemma 3.1. *Consider the edge $X_i \to X_j$ in the true causal graph. The orientation parameter θ_{ij} converges to $\sigma(\theta_{ij}) = 1$ if there exists a set of nodes $\widehat{pa}(X_j)$, for which the probability to be sampled as parents of X_j is greater than 0 and*

$$\mathbb{E}_{I_{X_i},\boldsymbol{X}}\left[\log p(X_j|\widehat{pa}(X_j), X_i) - \log p(X_j|\widehat{pa}(X_j))\right] > 0 \qquad (6)$$

The condition ensures that for an edge $X_i \to X_j$ in the true causal graph, the gradient of Equation 4 is negative for θ_{ij}, *i.e.* increases θ_{ij}. The intuition and proof behind this lemma is that when intervening on X_i, the distribution $p(X_j|...)$ remains unchanged and, hence, conditioning on X_i gives a better or equal log-likelihood estimate than without. In the opposite direction, it can only lead to positive gradients, *i.e.*, decreases θ_{ji}. With a similar derivation, we can also show that the orientation parameters are also correctly learned for ancestor-descendant pairs without a direct edge.

Using the fact that the orientation parameters will converge for parent-child and ancestor-descendant pairs, we can discuss the conditions for γ_{ij}. This requires us to make a comparison between the log-likelihood improvements and the sparsity regularizer, because setting the regularizer to $\lambda_{\text{sparse}} = \infty$ intuitively leads to an empty graph being optimal thus not converging to the intended graph.

Lemma 3.2. *Consider an edge $X_i \to X_j$ in the true causal graph. The parameter γ_{ij} converges to $\sigma(\gamma_{ij}) = 1$ if the following condition holds:*

$$\min_{\hat{pa}\subseteq gpa_i(X_j)} \mathbb{E}_{\hat{I}\sim p_I(I)}\mathbb{E}_{\tilde{p}_{\hat{I}}(\boldsymbol{X})}\left[\log p(X_j|\hat{pa}, X_i) - \log p(X_j|\hat{pa})\right] > \lambda_{sparse} \qquad (7)$$

where $gpa_i(X_j)$ is the set of nodes excluding X_i which, according to the ground truth graph, could have an edge to X_j without introducing a cycle.

In other words, we will recover an edge if the dependency of X_j on X_i in terms of log-likelihood cannot be replaced by other non-descendant variables up to the constant λ_{sparse}. Similar to the orientation parameters, this condition ensures that the gradients of γ_{ij} will be negative in expectation, *i.e.*, increases γ_{ij}, independent of all other values of $\boldsymbol{\gamma}$. We can exclude children and descendants of X_j in $\text{gpa}_i(X_j)$ because the likelihood of the orientation for these edges converges to zero according to Lemma 3.1. The condition can also be restricted to interventional data on X_i only instead of all variables if such is solely used to train γ_{ij}. However, in practice, less interventional data is available increasing the chances of inaccurate estimates, and sparser updates lead to slower training.

Finally, we discuss the convergence of all other edges:

Lemma 3.3. *If for all edges $X_i \to X_j$ in the true causal graph, $\sigma(\theta_{ij})$ and $\sigma(\gamma_{ij})$ have converged to one, the likelihood of all other edges will converge to zero.*

If all edges in the ground truth graph have converged, all other edges will have a gradient towards decreasing γ_{ij} as long as $\lambda_{\text{sparse}} > 0$. This is because all other pairs of variables are (conditionally) independent in the graph. This excludes children and descendants, but as discussed before, the orientation parameters converge to the opposite direction setting those edge likelihoods to zero.

From Lemma 3.2, we see that the lower we set λ_{sparse}, the more edges we can guarantee to discover. The trade-off is that low values of λ_{sparse} require longer training times since we need more graph and data samples in the gradient estimation. To summarize, we can give a guarantee for converging to the correct causal graph if for every edge $X_i \to X_j$ in the ground truth, the variables are not conditionally independent in the interventional data of X_i, and the minimal improvement on the log-likelihood estimate when conditioning on X_i is larger than λ_{sparse}.

3.5 Handling latent confounders

So far, we have assumed that all variables of the graph are observable and can be intervened on. A common issue in causal discovery is the existence of latent confounders, *i.e.*, an unobserved common cause of two or more variables introducing dependencies between each other. This leads to structure learning methods such as ENCO predicting false positive edges between variables with latent confounders. However, latent confounders for two variables X_i, X_j cause a unique pattern of learned parameters in ENCO: when intervening on X_i or X_j, having an edge between the two variables is disadvantageous, as in the intervened graph X_i and X_j are (conditionally) independent. For interventions on all other variables, an edge can be beneficial as X_i and X_j are correlated.

Using this characteristic, we extend ENCO to detect latent confounders. We restrict our discussion to latent confounders between two variables that do not have any direct edges with each other, and assume that the confounder is not a child of any other observed variable. We can still rely on the guarantees as outlined in Section 3.4 for all other edges besides between X_i and X_j. The reason is that Equation 7 already includes the possibility of additional edges in such cases. After convergence, we can score every pair of variables on how likely they share a latent confounder using a function $\text{lc}(\cdot)$ that is maximized in the scenario mentioned above. As edges are disadvantageous for the log-likelihood estimate when intervening on X_i or X_j but beneficial otherwise, we record the two gradients separately. In other words, we define $\gamma_{ij} = \gamma_{ij}^{(I)} + \gamma_{ij}^{(O)}$ where $\gamma_{ij}^{(I)}$ is only updated with gradients from Equation 3 under interventions on X_i, and $\gamma_{ij}^{(O)}$ on all others. With this separation, we define the following score function which is maximized for latent confounder between X_i and X_j:

$$\text{lc}(X_i, X_j) = \sigma\left(\gamma_{ij}^{(O)}\right) \cdot \sigma\left(\gamma_{ji}^{(O)}\right) \cdot \left(1 - \sigma\left(\gamma_{ij}^{(I)}\right)\right) \cdot \left(1 - \sigma\left(\gamma_{ji}^{(I)}\right)\right) \tag{8}$$

To converge to the mentioned values, especially of $\gamma_{ij}^{(O)}$, we need a similar condition as in Equation 7: the improvement on the log-likelihood estimate gained by the edge $X_i \rightarrow X_j$ and conditioned on all other parents of X_j needs to be larger than λ_{sparse} on interventional data excluding X_i and X_j. If this is not the case, the sparsity regularizer will instead remove the edge between X_i and X_j preventing any false predictions among observed variables. For all other pairs of variables, at least one of the terms in Equation 8 converges to zero. Thus, we can detect latent confounders by checking whether the score function $\text{lc}(X_i, X_j)$ is greater than a threshold hyperparameter $\tau \in (0.0, 1.0)$. We discuss possible guarantees in Appendix B, and experimentally verify this approach in Section 4.4.

4 Experiments

We evaluate ENCO on structure learning on synthetic datasets for systematic comparisons and real-world datasets for benchmarking against other methods in the literature. The experiments thereby focus on graphs with categorical variables. Categorical data is commonly more difficult in structure learning as regression techniques or assumption on linear noise models cannot be used. Yet, ENCO is also applicable on continuous data by using Normalizing Flows [35], for example. Our code is publicly available at `https://github.com/phlippe/ENCO`.

4.1 Experimental setup

Graphs and dataset Given a ground-truth causal graphical model, we sample 100k observational data points and 10k data points per intervention on each variable. Based on this data, ENCO and the baseline methods are tasked to recover the original DAG. For an intervention, the ground-truth conditional distribution is replaced by a uniform distribution for an intervened variable. In case of synthetic graphs, we follow the setup of Ke et al. [18] and create the ground-truth conditional distributions from neural networks. These networks take as input the categorical values of its variable's parents, and are initialized orthogonally to output a non-trivial distribution.

Hyperparameters ENCO We use two-layer MLPs with a final softmax for modeling the conditional distributions of each variable. In every epoch, we perform 1,000 update steps on the neural networks in distribution fitting and 100 updates on the graph parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$ in graph fitting. Overall, we train for 30 epochs. For estimating the gradients in graph fitting, we use $K = 100$ graph samples which is sufficient even for large graphs. The sparsity regularizer is set to $\lambda_{\text{sparse}} = 0.004$. More ablation studies on the hyperparameters and a detailed overview can be found in Appendix C.

Baselines We compare ENCO to GIES [13] and IGSP [47, 49] as greedy score-based approaches, and DCDI [4] and SDI [18] as continuous optimization methods. Pure constraint-based methods do not scale well to the given graph and dataset sizes [12, 33]. We do not compare to methods with

Table 1: Comparing structure learning methods in terms of structural hamming distance (SHD) on common graph structures (lower is better), averaged over 25 graphs each. In line with the theoretical guarantees, ENCO can reliably recover five out of the six graph structures without errors.

Graph type	bidiag	chain	collider	full	jungle	random
GIES [13]	47.4 (±5.2)	22.3 (±3.5)	13.3 (±3.0)	152.7 (±12.0)	53.9 (±8.9)	86.1 (±12.0)
IGSP [47]	33.0 (±4.2)	12.0 (±1.9)	23.4 (±2.2)	264.6 (±7.4)	38.6 (±5.7)	76.3 (±7.7)
SDI [18]	2.1 (±1.5)	0.8 (±0.9)	14.7 (±4.0)	121.6 (±18.4)	1.8 (±1.6)	1.8 (±1.9)
DCDI [4]	3.7 (±1.5)	4.0 (±1.3)	**0.0** (±0.0)	2.8 (±2.1)	1.2 (±1.5)	2.2 (±1.5)
ENCO (Ours)	**0.0** (±0.0)	**0.0** (±0.0)	**0.0** (±0.0)	**0.3** (±0.9)	**0.0** (±0.0)	**0.0** (±0.0)

observational data only since those can just recover the graph up to its Markov equivalence class. We perform a separate hyperparameter search for all baselines. Since SDI and DCDI use neural networks to fit (observational) distributions as well, we use the same network setup as for ENCO. All methods were executed on the same hardware, namely a 12-core CPU with a single NVIDIA RTX3090 GPU.

4.2 Causal structure learning on common graph structures

We first experiment on synthetic graphs for which we pick six common graph structures. The graphs `chain` and `full` represent the minimally- and maximally-connected DAGs. `bidiag` is a chain with 2-hop connections, and `jungle` is a tree-like graph. In the `collider` graph, one node has all other nodes as parents. Finally, `random` has a randomly sampled graph structure with a likelihood of 0.3 of two nodes being connected by a direct edge. For each graph structure, we generate 25 graphs with 25 nodes each on which we report the average performance and standard deviation. Following common practice, we use structural hamming distance (SHD) as evaluation metric. SHD counts the number of edges that need to be removed, added, or flipped in order to obtain the ground truth graph.

The results are summarized in Table 1. Overall, the continuous optimization methods considerably outperform the greedy search approaches. SDI works reasonably well on sparse graphs but struggles with nodes that have many parents. The second best is DCDI which performs well on the `collider` graph since its edges can be independently orientated. Although DCDI converges to acyclic graphs, it predicts some incorrectly oriented edges, while being 8 times slower than ENCO on the same hardware. ENCO reliably learns five out of six graph structures without errors except of rare mistakes on the `full` graph. Therefore, the theoretical guarantees also hold in practice for small graphs.

4.3 Scalability

Next, we test ENCO on graphs with large sets of variables. We create `random` graphs ranging from $N = 100$ to $N = 1,000$ nodes. Every node has on average 8 in- or outgoing edges and a maximum of 10 parents. The challenge of large graphs is that the number of possible edges grows quadractically and the number of DAGs super-exponentially. Hence, efficient methods are needed.

We compare ENCO to the two best performing baselines from Table 1, SDI [18] and DCDI [4]. All methods were given the same setup of neural networks and a maximum runtime which corresponds to 30 epochs for ENCO. We plot the SHD over graph size and runtime in Figure 3. ENCO is capable of recovering the causal graphs perfectly with no errors except for the $1,000$-node graph, for which it misses only one out of 1 million edges in 2 out of 10 experiments. SDI and DCDI achieve considerably worse performance. This shows that ENCO can efficiently be applied to $1,000$ variables

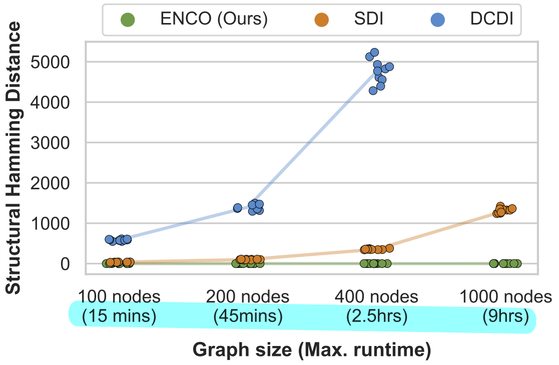


Figure 3: Evaluating SDI [18], DCDI [4], and ENCO on large graphs in terms of SHD (lower is better). Dots represent single experiments, lines connect the averages. DCDI ran out of memory for 1000 nodes.

Table 3: Results on graphs from the BnLearn library measured in structural hamming distance (lower is better). Results are averaged over 5 seeds with standard deviations listed in Appendix C.4. Despite deterministic variables and rare events, ENCO can recover all graphs with almost no errors.

Dataset	cancer [20] (5 nodes)	asia [21] (8 nodes)	sachs [37] (11 nodes)	child [41] (20 nodes)	alarm [2] (37 nodes)	diabetes [1] (413 nodes)	pigs [40] (441 nodes)
SDI [18]	3.0	4.0	7.0	11.8	24.6	422.4	18.0
ENCO (Ours)	**0.0**	**0.0**	**0.0**	**0.0**	**1.0**	**2.0**	**0.0**

while maintaining its convergence guarantees. For even larger graphs, prior knowledge of variables being independent can considerably increase efficiency by reducing the number of network inputs.

4.4 Detecting latent confounders

To test the detection of latent confounders, we create a set of 25 `random` graphs with 5 additional latent confounders. We make sure that every latent confounder is a parent of two randomly sampled nodes without a direct edge. The dataset is generated in the same way as before, except that we do not intervene on the latent confounders and remove them from the input data to ENCO. After training, we predict the existence of a latent confounder on any pair of variables X_i and X_j if $\text{lc}(X_i, X_j)$ is greater than τ. We choose $\tau = 0.4$ but verify in Appendix C.3 that the method is not sensitive to the specific value of τ. As shown in Table 2, ENCO detects more than 95% of the latent confounders without any false positives. What is more, the few mistakes do not affect the detection of all other edges which are recovered perfectly.

Table 2: Results of ENCO on detecting latent confounders averaged over 25 graphs with 25 nodes. The few missed confounders do not affect any other edge predictions.

Metrics	ENCO
SHD	0.0 (±0.0)
Confounder recall	96.8% (±9.5%)
Confounder precision	100.0% (±0.0%)

chain graph work also need this exp.

4.5 Real-world inspired data

Finally, we evaluate ENCO on a collection of causal graphs from the Bayesian Network Repository (BnLearn) [40]. The repository contains graphs inspired by real-world applications that are used as benchmarks in literature. In comparison to the synthetic graphs, the real-world graphs are sparser with a maximum of 6 parents per node and contain nodes with strongly peaked marginal distributions. They also contain deterministic variables, making the task challenging even for small graphs.

We compare ENCO to the best baseline from Sections 4.2 and 4.3, SDI [18], and evaluate the methods on 7 graphs with increasing sizes, see Table 3. We observe that ENCO recovers almost always the real-world causal graphs with perfect accuracy no matter their size. In contrast, SDI suffers from more mistakes as the graphs become larger. An exception is the `pigs` graph [40], which is very sparsely connected (a maximum of 2 parents per node), and hence easier to learn. The most challenging graph is `diabetes` [1] due to its large size and large amount of deterministic variables. ENCO makes only two mistakes showing that it can handle deterministic variables well. On the `alarm` graph [2], ENCO consistently misses out one edge. While using a lower regularizer can guarantee a perfect recovery, it would require more computation time and data. Yet, it is feasible. We conclude that ENCO can reliably perform structure learning on a wide variety of settings including deterministic variables.

5 Conclusion

In this paper, we propose ENCO, an efficient structure learning method leveraging observational and interventional data. ENCO models a graph by independent edge likelihoods with the edge orientation as a separate parameter. As such, ENCO converges to the correct causal graph under mild conditions. Its objective is unconstrained with respect to acyclicity allowing for low-variance gradient estimators. In experiments, we show that ENCO can be efficiently applied to graphs comprising hundreds of nodes, while also handling deterministic variables and possible latent confounders.

Limitations of ENCO include the need for interventional data on all variables. Future work includes investigating the generalization of ENCO to incomplete intervention sets. For instance, despite the absence of interventions one can still recover undirected edges via γ. A second limitation is that the orientations are missing transitivity: if $X_1 \succ X_2$ and $X_2 \succ X_3$, then $X_1 \succ X_3$ must also be true. Global order distributions such as Plackett-Luce [23, 34] require high variance gradient estimators and struggled with chains in early experiments. That said, a potential direction is to experiment with transitive relations for improving convergence speed and working on incomplete intervention sets.

Acknowledgments and Disclosure of Funding

Funding in direct support of this work: Qualcomm Technology, Inc. We thank Pascal Mettes and Christina Winkler for their valuable comments and feedback on an initial draft of this work, and Sara Magliacane for pointing out related works. We thank SURFsara for the support in using the Lisa Compute Cluster.

References

[1] Steen Andreassen, Roman Hovorka, Jonathan Benn, Kristian G. Olesen, and Ewart R. Carson. 1991. A model-based approach to insulin adjustment. In *Lecture Notes in Medical Informatics*, volume 44 of *Lecture Notes in Medical Informatics*, pages 239–249, Germany. Springer. Conference date: 24-06-1991 Through 27-06-1991.

[2] Ingo A. Beinlich, Henri Jacques Suermondt, R. Martin Chavez, and Gregory F. Cooper. 1989. The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *AIME 89*, pages 247–256, Berlin, Heidelberg. Springer Berlin Heidelberg.

[3] Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Nan Rosemary Ke, Sebastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher Pal. 2019. A meta-transfer objective for learning to disentangle causal mechanisms. *arXiv preprint arXiv:1901.10912*.

[4] Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. 2020. Differentiable causal discovery from interventional data. In *Advances in Neural Information Processing Systems*, volume 33, pages 21865–21877. Curran Associates, Inc.

[5] Heinze-Deml Christina, Meinshausen Nicolai, and Peters Jonas. 2018. Invariant Causal Prediction for Nonlinear Models. *Journal of Causal Inference*, 6(2):1–35.

[6] Thomas M. Cover and Joy A. Thomas. 2005. *Elements of Information Theory*. John Wiley and Sons, Ltd.

[7] A. Dixit, O. Parnas, B. Li, J. Chen, C. P. Fulco, L. Jerby-Arnon, N. D. Marjanovic, D. Dionne, T. Burks, R. Raychowdhury, B. Adamson, T. M. Norman, E. S. Lander, J. S. Weissman, N. Friedman, and A. Regev. 2016. Perturb-Seq: Dissecting Molecular Circuits with Scalable Single-Cell RNA Profiling of Pooled Genetic Screens. *Cell*, 167(7):1853–1866.

[8] Frederick Eberhardt. 2008. Almost optimal intervention sets for causal discovery. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, UAI'08, page 161–168, Arlington, Virginia, USA. AUAI Press.

[9] Frederick Eberhardt, Clark Glymour, and Richard Scheines. 2005. On the Number of Experiments Sufficient and in the Worst Case Necessary to Identify All Causal Relations among N Variables. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, page 178–184, Arlington, Virginia, USA. AUAI Press.

[10] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. 2000. Using bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4):601–620.

[11] Olivier Goudet, Diviyan Kalainathan, Philippe Caillou, Isabelle Guyon, David Lopez-Paz, and Michèle Sebag. 2017. Causal generative neural networks. *arXiv preprint arXiv:1711.08936*.

[TODO: Review] [12] Ruocheng Guo, Lu Cheng, Jundong Li, P. Richard Hahn, and Huan Liu. 2020. A survey of learning causality with data: Problems and methods. *ACM Comput. Surv.*, 53(4).

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

[14] Christina Heinze-Deml, Nicolai Meinshausen, and Jonas Peters. 2018. Invariant causal prediction for nonlinear models. *Journal of Causal Inference*, 6(2).

[15] John Hicks et al. 1980. *Causality in economics*. Australian National University Press.

[16] Antti Hyttinen, Frederick Eberhardt, and Matti Järvisalo. 2014. Constraint-based causal discovery: Conflict resolution with answer set programming. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, page 340–349, Arlington, Virginia, USA. AUAI Press.

[17] Oleg Ivanov, Michael Figurnov, and Dmitry Vetrov. 2019. Variational autoencoder with arbitrary conditioning. In *International Conference on Learning Representations*.

[18] Nan Rosemary Ke, Olexa Bilaniuk, Anirudh Goyal, Stefan Bauer, Hugo Larochelle, Chris Pal, and Yoshua Bengio. 2019. Learning neural causal models from unknown interventions. *arXiv preprint arXiv:1910.01075*.

[19] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[20] Kevin B. Korb and Ann E. Nicholson. 2010. *Bayesian artificial intelligence, second edition*. CRC Press, Australia.

[21] S. Lauritzen and D. Spiegelhalter. 1988. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the royal statistical society series b-methodological*, 50:415–448.

[22] Yang Li, Shoaib Akbar, and Junier Oliva. 2020. ACFlow: Flow models for arbitrary conditional likelihoods. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5831–5841. PMLR.

[23] R Duncan Luce. 1959. *Individual choice behavior*. John Wiley, Oxford, England.

[24] E. Z. Macosko, A. Basu, R. Satija, J. Nemesh, K. Shekhar, M. Goldman, I. Tirosh, A. R. Bialas, N. Kamitaki, E. M. Martersteck, J. J. Trombetta, D. A. Weitz, J. R. Sanes, A. K. Shalek, A. Regev, and S. A. McCarroll. 2015. Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets. *Cell*, 161(5):1202–1214.

[25] Ashique Rupam Mahmood, Hado Van Hasselt, and Richard S Sutton. 2014. Weighted importance sampling for off-policy learning with linear function approximation. In *NIPS*, pages 3014–3022.

[26] Christopher Meek. 1997. *Graphical Models: Selecting causal and statistical models*. Ph.D. thesis, Carnegie Mellon University.

[27] Ricardo Pio Monti, Kun Zhang, and Aapo Hyvärinen. 2020. Causal discovery with general non-linear relationships using non-linear ica. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 186–195. PMLR.

[28] Rainer Opgen-Rhein and Korbinian Strimmer. 2007. From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. *BMC Systems Biology*, 1(1):37.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

[30] Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[31] Judea Pearl. 2009. *Causality: Models, Reasoning and Inference*, 2nd edition. Cambridge University Press, USA.

[32] Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. 2016. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, pages 947–1012.

[TODO: Book] [33] Jonas Peters, Dominik Janzing, and Bernhard Schlkopf. 2017. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press.

[34] R. L. Plackett. 1975. The analysis of permutations. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 24(2):193–202.

[35] Danilo Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France. PMLR.

[36] J.M. Robins, M.A. Hernan, and B Brumback. 2000. Marginal structural models and causal inference in epidemiology. *Epidemiology*, 11(5):550–560.

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

[38] Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. 2012. On causal and anticausal learning. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 459–466, Madison, WI, USA. Omnipress.

[39] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. 2021. Towards Causal Representation Learning. *arXiv preprint arXiv:2102.11107*.

[40] Marco Scutari. 2010. Learning bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22.

[41] D. J. Spiegelhalter and R. G. Cowell. 1992. Learning in probabilistic expert systems. *Bayesian Statistics*, 4:447–466.

[42] Peter Spirtes, Clark Glymour, and Richard Scheines. 2000. *Causation, Prediction, and Search, Second Edition*. Adaptive computation and machine learning. MIT Press.

[43] Xiaohai Sun, Dominik Janzing, Bernhard Schölkopf, and Kenji Fukumizu. 2007. A kernel-based causal learning algorithm. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 855–862, New York, NY, USA. Association for Computing Machinery.

[44] Sofia Triantafillou and Ioannis Tsamardinos. 2015. Constraint-based causal discovery from multiple interventions over overlapping variable sets. *Journal of Machine Learning Research*, 16(66):2147–2205.

[45] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. 2006. The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm. *Machine Learning*, 65(1):31–78.

[46] Jan P Vandenbroucke, Alex Broadbent, and Neil Pearce. 2016. Causality and causal inference in epidemiology: the need for a pluralistic approach. *International journal of epidemiology*, 45(6):1776–1786.

[47] Yuhao Wang, Liam Solus, Karren Yang, and Caroline Uhler. 2017. Permutation-based causal inference algorithms with interventions. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[48] Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, pages 229–256.

[49] Karren Yang, Abigail Katcoff, and Caroline Uhler. 2018. Characterizing and learning equivalence classes of causal DAGs under interventions. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5541–5550. PMLR.

[50] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. 2018. GAIN: Missing data imputation using generative adversarial nets. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5689–5698. PMLR.

[51] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. 2019. DAG-GNN: DAG structure learning with graph neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7154–7163. PMLR.

[52] Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. 2018. Dags with no tears: Continuous optimization for structure learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9492–9503, Red Hook, NY, USA. Curran Associates Inc.

[53] Xun Zheng, Chen Dan, Bryon Aragam, Pradeep Ravikumar, and Eric Xing. 2020. 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*, pages 3414–3425. PMLR.

[54] Shengyu Zhu, Ignavier Ng, and Zhitang Chen. 2020. Causal discovery with reinforcement learning. In *International Conference on Learning Representations*.

Supplementary material
Efficient Neural Causal Discovery without Acyclicity Constraints

Table of Contents

A Gradient estimators

The following section describes in detail the derivation of the gradient estimators discussed in Section 3.3. We consider the problem of causal structure learning where we parameterize the graph by edge existence parameters $\boldsymbol{\gamma}$ and orientation parameters $\boldsymbol{\theta}$. Our objective is to optimize $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$ such that we maximize the probability of interventional data, *i.e.*, data generated from the true graphs under (arbitrary) interventions on single variables. Thereby, the likelihood estimates have been trained on observational data only. Additionally, we want to ensure that the graph is as sparse as possible to prevent unnecessary connections. Thus, an ℓ_1 regularizer is added on top of the edge probabilities. The full objective can be written as follows:

$$\tilde{\mathcal{L}} = \mathbb{E}_{\hat{I}\sim p_I(I)}\mathbb{E}_{\tilde{p}_{\hat{I}}(\boldsymbol{X})}\mathbb{E}_{p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C)}\left[\sum_{i=1}^{N}\mathcal{L}_C(X_i)\right] + \lambda_{\text{sparse}}\sum_{i=1}^{N}\sum_{j=1}^{N}\sigma(\gamma_{ij})\sigma(\theta_{ij}) \tag{9}$$

where:

- N is the number of variables in the causal graph $(X_1, ..., X_N)$;
- $p_I(I)$ is the distribution over interventions that are performed. This distribution can be set as a hyperparameter to weight certain interventions higher than others. In our experiments, we assume it to be uniform across interventions on variables;

- $\tilde{p}_{\hat{I}}(\boldsymbol{X})$ is the joint distribution of all variables under the intervention $\hat{I}$;
- $p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C)$ is the distribution over adjacency matrices C, which we model as a product of independent edge probabilities: $p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C) = \prod_{i=1}^{N} \prod_{j=1, j \neq i}^{N} \sigma(\gamma_{ij}) \cdot \sigma(\theta_{ij})$;
- $\mathcal{L}_C(X_i)$ is the negative log-likelihood estimate of variable X_i under sampled adjacency matrix C: $\mathcal{L}_C(X_i) = -\log f_{\phi_i}(X_i; C_{\cdot,i} \odot \boldsymbol{X}_{-i})$;
- λ_{sparse} is a hyperparameter representing the regularization weight.

Based on this objective, we derive the gradient estimators for optimizing both edge existence and orientation parameters.

A.1 Low-variance gradient estimator for edge parameters

In order to optimize the edge parameters via SGD, we need to determine the gradient $\frac{\partial}{\partial \gamma_{ij}} \tilde{\mathcal{L}}$. Since $\mathcal{L}$ consists of a sum of two terms, *i.e.*, the log-likelihood estimate and the regularization, we can look at both parts separately. To prevent any confusion of index variables, we will use k, l as indices for the parameter γ_{kl} for which we determine the gradient, *i.e.*, $\frac{\partial}{\partial \gamma_{kl}} \tilde{\mathcal{L}}$, and i, j as indices for sums.

As a first step, we ① determine the gradients for the regularization term. Those can be found by taking the derivative of the sigmoid:

$$\frac{\partial}{\partial \gamma_{kl}} \lambda_{\text{sparse}} \sum_{i=1}^{N} \sum_{j=1}^{N} \sigma(\gamma_{ij}) \sigma(\theta_{ij}) = \sigma(\gamma_{kl}) \cdot (1 - \sigma(\gamma_{kl})) \cdot \sigma(\theta_{kl}) \lambda_{\text{sparse}} \tag{10}$$

Thus, it is straight-forward to calculate for any edge parameter. In the following, we use $\sigma'(...)$ to abbreviate the derivate of the sigmoid: $\sigma'(\gamma_{kl}) = \sigma(\gamma_{kl})(1 - \sigma(\gamma_{kl}))$.

② For the log-likelihood term, we start by reorganizing the expectations to simplify the gradient expression. The derivate term $\frac{\partial}{\partial \gamma_{kl}}$ can be moved inside the two expectations over interventional data since those are independent of the graph parameters. Thus, we can write:

$$\frac{\partial}{\partial \gamma_{kl}} \tilde{\mathcal{L}}' = \mathbb{E}_{\hat{I} \sim p_I(I)} \mathbb{E}_{\tilde{p}_{\hat{I}}(\boldsymbol{X})} \frac{\partial}{\partial \gamma_{kl}} \mathbb{E}_{p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C)} \left[\sum_{i=1}^{N} \mathcal{L}_C(X_i) \right] \tag{11}$$

For readability, we denote $\tilde{\mathcal{L}}'$ to be the objective in Equation 9 without the regularizer.

Next, we take a closer look at the derivate of the expectation over adjacency matrices. Note that we have defined the adjacency matrix distribution as $p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C) = \prod_{i=1}^{N} \prod_{j=1, j \neq i}^{N} \sigma(\gamma_{ij}) \sigma(\theta_{ij})$, with $C_{ij} = 1$ representing the edge $X_i \to X_j$. Since a parameter γ_{ij} only influences the likelihood of the edge $X_i \to X_j$ and no other edges, we can reduce the expectation to a single binary variable over which we need to differentiate the expectation:

$$\frac{\partial}{\partial \gamma_{kl}} \mathbb{E}_{p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C)} \left[\sum_{i=1}^{N} \mathcal{L}_C(X_i) \right] = \mathbb{E}_{p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C_{-kl})} \left[\frac{\partial}{\partial \gamma_{kl}} \mathbb{E}_{p_{\gamma,\theta}(C_{kl})} \left[\sum_{i=1}^{N} \mathcal{L}_C(X_i) \right] \right] \tag{12}$$

where $p_{\gamma,\theta}(C_{kl}) = \sigma(\gamma_{kl}) \cdot \sigma(\theta_{kl})$. The first expectation over $p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C_{-kl})$ is independent of γ_{kl} as we have defined the adjacency matrix distribution to be a product of independent edge probabilities.

The log-likelihood estimate of a variable, $\mathcal{L}_C(X_i)$, depends on the adjacency matrix column $C_{\cdot,i}$ which represents the input connections to the node X_i. All other edges have no influence on the log-likelihood estimate of X_i. Hence, the parameter γ_{kl} only influences $\mathcal{L}_C(X_l)$, and thus we can reduce the sum inside the expectation to:

$$\frac{\partial}{\partial \gamma_{kl}} \mathbb{E}_{p_{\gamma,\theta}(C_{kl})} \left[\sum_{i=1}^{N} \mathcal{L}_C(X_i) \right] = \frac{\partial}{\partial \gamma_{kl}} \mathbb{E}_{p_{\gamma,\theta}(C_{kl})} \left[\mathcal{L}_C(X_l) \right] \tag{13}$$

The REINFORCE trick is a simple method to move the derivative of a discrete distribution inside the expectation. Applied to our situation, we obtain:

$$\frac{\partial}{\partial \gamma_{kl}} \mathbb{E}_{p_{\gamma,\theta}(C_{kl})} \left[\mathcal{L}_C(X_l) \right] = \mathbb{E}_{p_{\gamma,\theta}(C_{kl})} \left[\mathcal{L}_C(X_l) \frac{\partial \log p_{\gamma,\theta}(C_{kl})}{\partial \gamma_{kl}} \right] \tag{14}$$

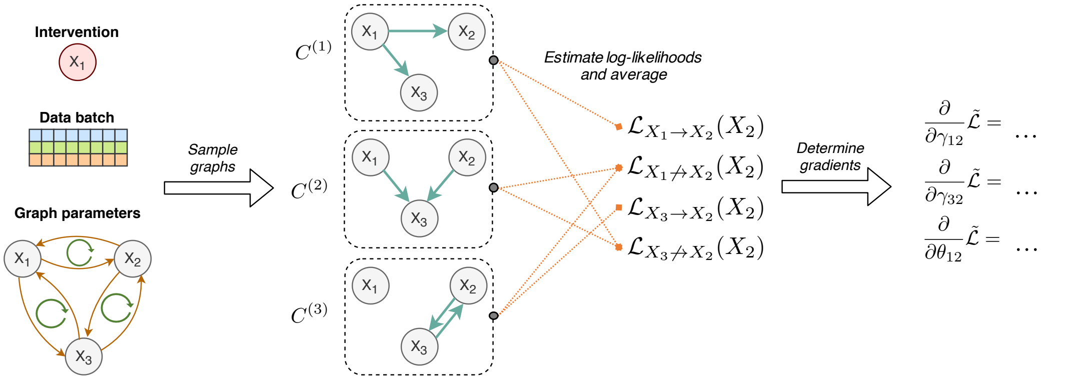

Intervention
X_1
Data batch
Graph parameters
X_1
X_2
X_3
Sample graphs
$C^{(1)}$
$C^{(2)}$
$C^{(3)}$
Estimate log-likelihoods and average
$\mathcal{L}_{X_1\to X_2}(X_2)$
$\mathcal{L}_{X_1\not\to X_2}(X_2)$
$\mathcal{L}_{X_3\to X_2}(X_2)$
$\mathcal{L}_{X_3\not\to X_2}(X_2)$
Determine gradients
$\frac{\partial}{\partial\gamma_{12}}\tilde{\mathcal{L}} = \dots$
$\frac{\partial}{\partial\gamma_{32}}\tilde{\mathcal{L}} = \dots$
$\frac{\partial}{\partial\theta_{12}}\tilde{\mathcal{L}} = \dots$

Figure 4: Visualizing the gradient calculation for the incoming edges of X_2 in an example graph with three variables. The intervention is being performed on X_1, and the data is used to calculate the log-likelihood estimates under the three randomly sampled graphs: $\mathcal{L}_{C^1}(X_2)$, $\mathcal{L}_{C^2}(X_2)$ and $\mathcal{L}_{C^3}(X_2)$. Those terms are assigned to the Monte-Carlo estimators for $\mathcal{L}_{X_i\to X_2}(X_2)$ and $\mathcal{L}_{X_i\not\to X_2}(X_2)$, and finally used to determine the gradients for $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$. The same process is performed for X_3 as well.

This leaves us with two cases in the expectation: $C_{kl} = 0$ and $C_{kl} = 1$. In other words, we need to distinguish between samples of C where we have the edge $X_k \to X_l$, and where we do not have such an edge ($X_k \not\to X_l$). Thus, we can also write the expectation as a weighted sum of those two cases:

$$\begin{aligned}\mathbb{E}_{p_{\gamma,\theta}(C)}\left[\mathcal{L}_C(X_l)\frac{\partial \log p_{\gamma,\theta}(C_{kl})}{\partial\gamma_{kl}}\right] &= \sigma(\gamma_{kl})\cdot\sigma(\theta_{kl})\cdot\mathcal{L}_{X_k\to X_l}(X_l)\cdot\frac{\partial\log\sigma(\gamma_{kl})\cdot\sigma(\theta_{kl})}{\partial\gamma_{kl}}+\\ &\qquad (1-\sigma(\gamma_{kl})\cdot\sigma(\theta_{kl}))\cdot\mathcal{L}_{X_l\not\to X_k}(X_k)\cdot\frac{\partial\log\left(1-\sigma(\gamma_{kl})\cdot\sigma(\theta_{kl})\right)}{\partial\gamma_{kl}}\end{aligned} \tag{15}$$

We use $\mathcal{L}_{X_k\to X_l}(X_l)$ to denote the (expected) negative log likelihood for X_l under adjacency matrices where we have an edge from X_k to X_l:

$$\mathcal{L}_{X_k\to X_l}(X_l) = \mathbb{E}_{p_{\gamma,\theta}(C_{-kl}),C_{kl}=1}\left[\mathcal{L}_C(X_l)\right] \tag{16}$$

$$\mathcal{L}_{X_k\not\to X_l}(X_l) = \mathbb{E}_{p_{\gamma,\theta}(C_{-kl}),C_{kl}=0}\left[\mathcal{L}_C(X_l)\right] \tag{17}$$

The final step is to solve the two derivative terms in Equation 14. This is done as follows:

$$\frac{\partial\log\sigma(\gamma_{kl})\cdot\sigma(\theta_{kl})}{\partial\gamma_{kl}} = \frac{\partial\log\sigma(\gamma_{kl})+\log\sigma(\theta_{kl})}{\partial\gamma_{kl}} = 1-\sigma(\gamma_{kl}) \tag{18}$$

$$\frac{\partial\log\left(1-\sigma(\gamma_{kl})\cdot\sigma(\theta_{kl})\right)}{\partial\gamma_{kl}} = -\frac{\sigma(\gamma_{kl})\cdot(1-\sigma(\gamma_{kl}))\cdot\sigma(\theta_{kl})}{1-\sigma(\gamma_{kl})\cdot\sigma(\theta_{kl})} \tag{19}$$

Putting these results back in the original equation and adding the sparsity regularizer, we get:

$$\frac{\partial}{\partial\gamma_{ij}}\tilde{\mathcal{L}} = \sigma(\gamma_{ij})\cdot(1-\sigma(\gamma_{ij}))\cdot\sigma(\theta_{ij})\cdot\mathbb{E}_{\boldsymbol{X},C_{-ij}}\left[\mathcal{L}_{X_i\to X_j}(X_j)-\mathcal{L}_{X_i\not\to X_j}(X_j)+\lambda_{\text{sparse}}\right] \tag{20}$$

To align the result with the gradient in Section 3.3, we switch the index notation from k, l to i, j again. The expectation $\mathbb{E}_{\boldsymbol{X},C_{-ij}}$ is a short form for the expectations $\mathbb{E}_{\hat{I}\sim p_I(I)}\mathbb{E}_{\tilde{p}_{\hat{I}}(\boldsymbol{X})}\mathbb{E}_{p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C_{-ij})}$. From this expression, we can see that the gradients of γ_{ij} are proportional to the difference of the expected negative log-likelihood of X_j with having an edge between $X_i \to X_j$, and the cases where $X_i \not\to X_j$. The sparsity regularizer thereby biases the difference towards the no-edge case. The value of γ_{ij} and θ_{ij} only scale the gradient, but do not influence its direction.

In order to train this objective on a dataset of interventional data, we can use Monte-Carlo sampling to obtain an unbiased gradient estimator. Note that the adjacency matrix samples to estimate $\mathcal{L}_{X_i\to X_j}(X_j)$ and $\mathcal{L}_{X_i\not\to X_j}(X_j)$ are not required to be the same. For efficiency, we instead sample K adjacency matrices from $p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C)$, evaluate the likelihood of a batch $\boldsymbol{X}$ under all these graphs. Afterwards, we assign the evaluated samples to one of the two cases, depending on C_{ij} being zero or

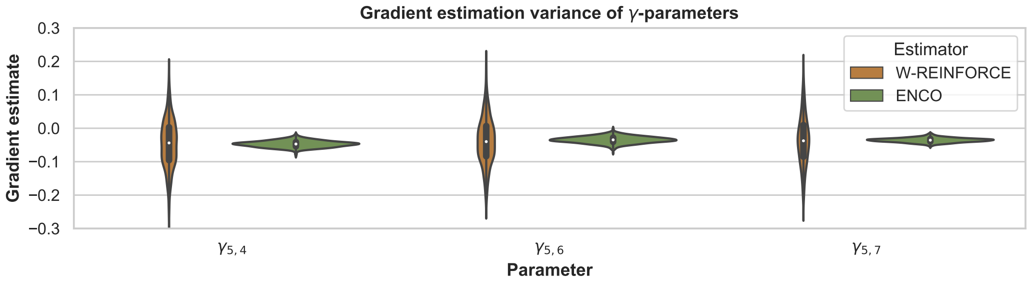


Figure 5: Plotting the gradient estimate variance of ENCO for three variables of γ compared to previous REINFORCE-like approach by Bengio et al. [3] on an example graph with $K = 100$. The gradients have been scaled to match in terms of averages. We can see a clear reduction in variance with the gradient estimator of ENCO allowing us to use lower sample sizes.

one. This way, we can reuse the same graph samples for all edge parameters $\boldsymbol{\gamma}$. We visualize the gradient calculation in Figure 4. In the cases where we perform an intervention on X_i, we do not optimize γ_{ij} for this step and set the gradients to zero. The same holds for gradient steps where we do not have any samples for one of the two log-likelihood estimates.

A.1.1 Comparison to previous gradient estimators

As discussed in Section 3, previous work on similar structure learning methods [3, 18] relied on a different estimator. In terms of derivation, the main difference is the continuation from Equation 14 on. In our proposed method, we write the expectation as the sum of two terms that can independently be approximated via Monte-Carlo sampling. In comparison, Bengio et al. [3] proposed to directly apply a Monte-Carlo sampler to Equation 14, and apply an importance sampling weight to reduce the variance. This estimator is also used in the method SDI [18] to which we have experimentally compared our method.

Figure 2 compared the gradient estimator in terms of standard deviation. The gradient estimator of ENCO achieves a 10 times lower standard deviation compared to Bengio et al. [3] making it much more efficient. Since the estimator by Bengio et al. [3] is biased and has a different mean, we have scaled both estimators to have the same mean. Specifically, we have applied ENCO to `random` graphs from our experiments on synthetic graphs (see Section 4.2), and evaluated $64,000$ sampled adjacency matrices in terms of log-likelihood estimates. These $64,000$ samples are grouped into sets of K samples which we have used to estimate the gradients of $\boldsymbol{\gamma}$. We evaluated different values of K, from $K = 20$ to $K = 4000$, and plotted the standard deviation of those estimates in Figure 2. We have also visualized three examples as violin plots in Figure 5 that demonstrate that despite both estimators having a similar mean, the variance of gradient estimates is much higher for Bengio et al. [3].

To verify that the improvement of ENCO is not just because of the gradient estimators, we have performed an ablation study with ENCO deploying the gradient estimator of Bengio et al. [3] in Appendix C.5.

A.2 Low-variance gradient estimator for orientation parameters

To derive the gradients for the orientation parameters $\boldsymbol{\theta}$, we can mostly follow the same approach as for the edge existence parameters $\boldsymbol{\gamma}$. However, we have to keep in mind the constraint $\theta_{kl} = -\theta_{lk}$ which ensures that the orientation probability sums to one: $\sigma(\theta_{kl}) + \sigma(\theta_{lk}) = 1$.

To determine the gradient of the likelihood term, we can separate the two gradients of θ_{kl} and θ_{lk}. This is because θ_{kl} only influences the expectation over $\mathcal{L}_C(X_l)$, while θ_{lk} concerns $\mathcal{L}_C(X_k)$. We can follow Equation 11 to Equation 20 of Section A.1 by swapping θ_{kl} and γ_{kl}. For the derivative through the expectation, we obtain the following gradient:

$$\frac{\partial}{\partial \theta_{kl}} \mathbb{E}_{p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C)}\left[\mathcal{L}_C(X_l)\right] = \sigma'(\theta_{kl}) \cdot \sigma(\gamma_{kl}) \cdot \mathbb{E}_{p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C_{-kl})}\left[\mathcal{L}_{X_k \to X_l}(X_l) - \mathcal{L}_{X_k \not\to X_l}(X_l)\right] \tag{21}$$

Since we have the condition that $\theta_{kl} = -\theta_{lk}$, the full gradient for θ_{kl} would therefore consist of the gradient above minus the gradient of Equation 21 with respect to θ_{ji}. However, as discussed in Section 3.3, the orientation of an edge cannot be learned from observational data in this framework. Hence, we only want to use the gradients of θ_{kl} if we intervene on node X_k, which gives us the following gradient expression:

$$\begin{aligned}\frac{\partial}{\partial\theta_{ij}}\tilde{\mathcal{L}} = \sigma'(\theta_{ij})\Big(&p(I_{X_i})\cdot\sigma(\gamma_{ij})\cdot\mathbb{E}_{I_{X_i},\boldsymbol{X},C_{-ij}}\left[\mathcal{L}_{X_i\to X_j}(X_j)-\mathcal{L}_{X_i\not\to X_j}(X_j)\right]- \\ &p(I_{X_j})\cdot\sigma(\gamma_{ji})\cdot\mathbb{E}_{I_{X_j},\boldsymbol{X},C_{-ij}}\left[\mathcal{L}_{X_j\to X_i}(X_i)-\mathcal{L}_{X_j\not\to X_i}(X_i)\right]\Big)\end{aligned} \tag{22}$$

To align the equation with the one in Section 3.3, we swap the indices k, l with i, j again. The first line represents cases where we have an intervention on the variable X_i, while we have it over interventions on the variable X_j in the second line. The two terms are weighted based on the edge existence likelihood $\sigma(\gamma_{ij})$ and $\sigma(\gamma_{ji})$ respectively, and the likelihood of performing an intervention on X_i or X_j. In our experiments, we use a uniform probability across interventions on variables, but emphasize that this is not strictly required. Moreover, one could design heuristics that selects the intervention to update the parameters on with the aim of increasing computational efficiency. The gradient estimators presented in Equation 22 would still be valid in such a case.

We clarify that we do not consider the gradients of θ_{ij} with respect to the edge regularizer. This is done for two reasons. Firstly, the orientation parameter models only the direction of the edge, not whether it exists or not. The regularizer would increase θ_{ij} if the edge existence for the opposite direction would be greater than for the direction from X_i to X_j, *i.e.*$\gamma_{ij} < \gamma_{ji}$, and decrease θ_{ij} if we have $\gamma_{ij} > \gamma_{ji}$. However, the orientation should only model the causal direction of an edge. Hence, we do not gain any value from a causal perspective when adding the regularizer to the gradient. Secondly, the regularizer would require us to take additional assumptions for guaranteeing the discovery of the true graph upon convergence. In experiments with using a regularizer in the θ-gradient, we did not observe any difference to the experiments without the regularizer.

A.3 Training loop

Finally, we give an overview over the full training loop in Algorithm 1. The distribution over interventions $p(I)$ is set to a uniform distribution for all our experiments. However, the distribution can also be replaced by a heuristic which selects interventions to increase computational efficiency. To keep the convergence guarantees, $p(I)$ would have to guarantee a non-zero probability to pick any variable. In experiments, we experienced that the Adam optimizer [19] speeds up the convergence of the parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$ while not interfering with the convergence guarantees in practice.

Algorithm 1: Learning algorithm of ENCO

```
Data: $N$ variables $\{X_1, ..., X_N\}$; observational dataset $D_{\text{obs}}$; interventional datasets $D_{\text{int}}(\hat{I})$ for
      sparse, perfect interventions on all variables; distribution over interventions $p(I)$
Result: A graph structure corresponding to the cuasal relations among variables
Initialize $\boldsymbol{\gamma} = \mathbf{0}, \boldsymbol{\theta} = \mathbf{0}$
for number of epochs do
    /* Distribution fitting                                              */
    for $F$ iterations do
        $\boldsymbol{X} \sim D_{\text{obs}}$
        for $i \leftarrow 1$ to $N$ do
            $M_l \sim \text{Ber}(\sigma(\gamma_{li}) \cdot \sigma(\theta_{li}))$
            $L = -\log f_{\phi_i}(X_i; \boldsymbol{M}_{-i} \odot \boldsymbol{X}_{-i})$
            $\phi_i \leftarrow \text{Adam}(\phi_i, \nabla_{\phi_i} L)$
        end
    end
    /* Graph fitting                                                     */
    for $G$ iterations do
        /* Sample an intervention                                        */
        $\hat{I} \sim p(I)$ with intervention target $X_t$
        $\boldsymbol{X} \sim D_{\text{int}}(\hat{I})$
        /* Evaluate multiple graph samples for gradient estimator        */
        for $k = 1$ to $K$ do
            $C^{(k)} \sim \text{Ber}(\sigma(\boldsymbol{\gamma}) \cdot \sigma(\boldsymbol{\theta}))$
            for $i = 1$ to $N$ do
                $\mathcal{L}_{C^{(k)}}(X_i) \leftarrow -\log f_{\phi_i}(X_i; C^{(k)}_{:,i} \odot \boldsymbol{X}_{-i})$
            end
        end
        /* Update parameters                                             */
        for $i = 1$ to $N$ do
            for $j = 1$ to $N$ where $j \neq i$ and $j \neq t$ do
                /* Considering edge $X_i \rightarrow X_j$                */
                $\mathcal{L}_{X_i \rightarrow X_j}(X_j) \leftarrow \frac{\sum_{k=1}^{K} C_{ij}^{(k)} \cdot \mathcal{L}_{C^{(k)}}(X_j)}{\sum_{k=1}^{K} C_{ij}^{(k)}}$
                $\mathcal{L}_{X_i \not\rightarrow X_j}(X_j) \leftarrow \frac{\sum_{k=1}^{K} (1-C_{ij}^{(k)}) \cdot \mathcal{L}_{C^{(k)}}(X_j)}{\sum_{k=1}^{K} (1-C_{ij}^{(k)})}$
                $\gamma_{ij} \leftarrow \gamma_{ij} - \sigma'(\gamma_{ij}) \cdot \sigma(\theta_{ij}) \cdot \left[\mathcal{L}_{X_i \rightarrow X_j}(X_j) - \mathcal{L}_{X_i \not\rightarrow X_j}(X_j) + \lambda_{\text{sparse}}\right]$
                if $i == t$ then
                    $\theta_{ij} \leftarrow \theta_{ij} - \sigma'(\theta_{ij}) \cdot \sigma(\gamma_{ij}) \cdot \left[\mathcal{L}_{X_i \rightarrow X_j}(X_j) - \mathcal{L}_{X_i \not\rightarrow X_j}(X_j)\right]$
                    $\theta_{ji} \leftarrow -\theta_{ij}$
                end
            end
        end
    end
end
return $G = (V, E)$ with $V = \boldsymbol{X}$ and
 $E = \{(X_i, X_j) \mid i, j \in [1, N]$ and $i \neq j$ and $\sigma(\gamma_{ij}) > 0.5$ and $\sigma(\theta_{ij}) > 0.5\}$
```

B Conditions for converging to the true causal graph

The following section gives an overview and proves the conditions under which ENCO converges to the correct causal graph given sufficient time and data. To make the proof more accessible, we will first discuss the assumptions that are needed for the guarantee, and then give a sketch of the proof.

B.1 Assumptions

Assumption 1 A common assumption in causal structure learning is that the data distribution over all variables $p(\boldsymbol{X})$ is Markovian and faithful with respect to the causal graph we are trying to model. This means that the graph represents the (conditional) independencies relations between variables in the data, and (conditional) independence relations in the data reflect the edges in the graph. For ENCO, faithfulness is not strictly required. This is because we work with interventional data. Instead, we rely on the Markov property and assume that for all variables, the parent set $\text{pa}(X_i)$ reflects the inputs to the causal generation mechanism of X_i. This allows us to also handle deterministic variables.

Assumption 2 For this proof, we assume that all variables of the graph are known and observable, and no latent confounders exist. Latent confounders can introduce dependencies between variables which are not reflected by the ground truth graph solely on the observed variables.

Assumption 3 ENCO relies on neural networks to determine the conditional data distributions $p(X_i|...)$. Hence, for providing a guarantee, we assume that in the graph learning step the neural networks have been sufficiently trained such that they accurately model all possible conditional distribution $p(X_i|...)$. In practice, the neural networks might have a slight error. However, as long as enough data, network complexity, and training time is provided, it is fair to assume that the difference between the modeled distribution and the true conditional is smaller than an arbitrary constant ϵ.

B.2 Convergence conditions

The proof of the convergence conditions consists of the following three main steps:

Step 1 We show under which conditions the orientation parameters θ_{ij} will converge to $+\infty$, i.e. $\sigma(\theta_{ij}) \to 1$, if X_i is an ancestor of X_j. Similarly, if X_j is an descendant of X_i, the parameter θ_{ij} will converge to $-\infty$, i.e. $\sigma(\theta_{ij}) \to 0$.

Step 2 Under the assumption that the orientation parameters have converged as in Step 1, we show that for edges in the true graph, γ_{ij} will converge to 1. Note that we need to take additional assumptions/conditions with respect to λ_{sparse} here.

Step 3 Once the parameters γ_{ij} and θ_{ij} have converged for the edges in the ground truth graph, we show that all other edges will be removed by the sparsity regularizer.

The following paragraphs provide more details for each step. Note that causal graphs that do not fulfill all parts of the convergence guarantee can still eventually be recovered. The reason is that the conditions listed in the theorems below ensure that there exists no local minima for $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$ to converge in. Although local minima exist, the optimization process might converge to the global minimum of the true causal graph.

Theorem B.1. *Consider the edge $X_i \to X_j$ in the true causal graph. The orientation parameter θ_{ij} converges to $\sigma(\theta_{ij}) = 1$ if the following two conditions are fulfilled:*

(1) for all possible sets of parents of X_j excluding X_i, adding X_i improves the log-likelihood estimate of X_j under the intervention on X_i, or leaves it unchanged:

$$\forall \widehat{pa}(X_j) \subseteq X_{-i,j} : \mathbb{E}_{I_{X_i},\boldsymbol{X}} \left[\log p(X_j|\widehat{pa}(X_j), X_i) - \log p(X_j|\widehat{pa}(X_j))\right] \geq 0 \tag{23}$$

(2) there exists a set of nodes $\widehat{pa}(X_j)$, for which the probability to be sampled as parents of X_j is greater than 0, and the following condition holds:

$$\exists \widehat{pa}(X_j) \subseteq X_{-i,j} : \mathbb{E}_{I_{X_i},\boldsymbol{X}} \left[\log p(X_j|\widehat{pa}(X_j), X_i) - \log p(X_j|\widehat{pa}(X_j))\right] > 0 \tag{24}$$

Proof. Based on the conditions in Equations 23 and 24, we need to show that the gradient of θ_{ij} is negative in expectation, independent of other values of $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$. For readability, we define the following function:

$$T(X_k, X_l) = \mathbb{E}_{I_{X_k},\boldsymbol{X},C_{-kl}}\left[\mathcal{L}_{X_k \to X_l}(X_l) - \mathcal{L}_{X_k \not\to X_l}(X_l)\right] \tag{25}$$

Hence, the gradient of θ_{ij} can be written as:

$$\frac{\partial}{\partial \theta_{ij}}\tilde{\mathcal{L}} = \sigma'(\theta_{ij}) \cdot \Big(p(I_{X_i}) \cdot \sigma(\gamma_{ij}) \cdot T(X_i, X_j) - p(I_{X_j}) \cdot \sigma(\gamma_{ji}) \cdot T(X_j, X_i)\Big) \tag{26}$$

Looking at the gradient of θ_{ij} in Equation 26, the conditions correspond to $T(X_i, X_j)$ being smaller or equals to zero. Note that the sign is flipped because in $T(X_i, X_j)$, we have negative log-likelihoods represented by $\mathcal{L}_{X_k \to X_l}(X_l)$, while in Equations 23 and 24, we have log-likelihoods. Further, the other factors of $\sigma'(\theta_{ij})$, $\sigma(\gamma_{ij})$ and $p(I_X)$ are all limited in the range of $(0, 1)$ meaning that the sign of the gradient is solely determined by $T(X_i, X_j)$ and $T(X_j, X_i)$. If $T(X_i, X_j) - T(X_j, X_i)$ is smaller than zero, then the gradient of θ_{ij} is negative, *i.e.*increasing θ_{ij}.

First, we look at when $T(X_i, X_j) < 0$. The condition in Equation 23 ensures that conditioning X_j on a true parent X_i when intervening on X_i does not lead to a worse log-likelihood estimate than without. While this condition might seem natural, there are special cases where this condition does not hold for all variables (see Section B.2.1). The second condition, Equation 24, guarantees that there is at least one parent set for which $T(X_i, X_j)$ is negative. Therefore, in expectation over all possible adjacency matrices $p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C)$, $T(X_i, X_j)$ is smaller than zero if the two conditions hold.

To guarantee that the whole gradient of θ_{ij} is negative, we also need to show that for interventions on X_j, $T(X_j, X_i)$ can only be positive. When intervening on X_j, X_i and X_j become independent as the edge $X_i \to X_j$ is removed in the intervened graph. A distribution $p(X_i|X_j, ...)$ relying on correlations between X_i and X_j from observational data cannot achieve a better estimate than the same distribution when removing X_j. This is because the cross entropy is minimized when the sampled distribution, in this case $p(X_i)$, is equal to the log-likelihood estimator [6]:

$$-\sum_{x_i,x_j} p(X_i = x_i) \log p(X_i = x_i) \leq \sum_{x_i,x_j} p(X_i = x_i) \log q(X_i = x_i | X_j = x_j) \tag{27}$$

The only situation where X_i and X_j can become conditionally dependent under interventions on X_j is if X_i and X_j share a collider X_k, and X_i is being conditioned on the collider X_k and X_j. However, this requires that θ_{ki} has negative gradients, *i.e.* θ_{ki} increasing, when intervening on X_k. This cannot be the case since under interventions on X_k, X_i and X_k become conditionally independent, and the correlations learned from observational data cannot be transferred to the interventional setting. If X_k and X_i again share a collider, we can apply this argumentation recursively until a node X_n does not share a collider with X_i. The recursion will always come to an end as we have a finite set of nodes, and the causal graph is assumed to be acyclic.

Thus, if the conditions in Equations 23 and 24 holds for all edges $X_i \to X_j$ in the causal graph, we can guarantee that with sufficient time and data, the corresponding orientation parameters θ_{ij} will converge to $\sigma(\theta_{ij}) = 1$. □

Theorem B.2. *Consider a pair of variables X_i, X_j for which X_i is an ancestor of X_j without direct edge in the true causal graph. Then, the orientation parameter of the edge $X_j \to X_i$ converges to $\sigma(\theta_{ij}) = 1$ if the same conditions as in Theorem B.1 hold for the pair of X_i, X_j.*

Proof. To show this theorem, we need to consider two cases for a pair of variables X_i and X_j: X_i and X_j are conditionally independent under a sampled adjacency matrix C, or X_i and X_j are not independent. Both cases need to be considered for an intervention on X_i with the log-likelihood estimate of X_j, and an intervention on X_j with the log-likelihood estimate of X_i.

First, we discuss interventions on X_i. If under the sampled adjacency matrix C, X_j is conditionally independent of X_i, the difference in the log-likelihood estimates $T(X_i, X_j)$ is zero in expectation. The variables can be independent if, for example, the parents of X_j are all parents of the true causal graph. If X_j is not conditionally independent of X_i, the conditions in Equations 23 and 24 from Theorem B.1 ensure that X_i has, in expectation, a positive effect on the log-likelihood estimate of X_j. Thus, under interventions on X_i, the gradient of θ_{ij} is smaller or equals to zero, *i.e.*increases θ_{ij}.

Next, we consider interventions on X_j. If under the sampled adjacency matrix X_i is conditionally independent of X_j, the difference in the log-likelihood estimates $T(X_j, X_i)$ is zero. The variables can be independent if X_i is conditioned on variables that d-separate X_i and X_j in the true causal graph. For instance, having the children of X_i as parents of X_i creates this scenario. However, for this scenario to take place, one or more orientation parameters of parent-child or ancestor-descendant pairs must be incorrectly converged. In case of a parent-child pair X_i, X_k, Theorem B.1 shows that $\sigma(\theta_{ik})$ will converge to one removing any possibility of a reversed edge to be sampled. In case of an ancestor-descendant pair X_i, X_l, we can apply a recursive argument: as X_l d-separates X_i and X_j, X_l must come before X_j in the causal order. If for the gradient θ_{il}, we have a similar scenario with X_i being conditionally independent of X_j, the same argument applies. This can be recursively applied until no more variables except direct children of X_i can d-separate X_i and X_j. In that case, $\sigma(\theta_{ik})$ will converge to one, which leads to all other orientation parameters to converge to one as well. If X_i is not conditionally independent of X_j, we can rely back on the argumentation of Theorem B.1 when we have an edge $X_i \to X_j$: as in the intervened causal graph, X_i and X_j are independent, any correlation learned from observational data can only lead to a worse log-likelihood estimate. In cases of colliders, we can rely on the recursive argument from before. Thus, under interventions on X_j, the gradient of θ_{ij} must be smaller or equals to zero in expectation, *i.e.*increases θ_{ij}.

Therefore, we can conclude that $\sigma(\theta_{ij})$ converges to one for any ancestor-descendant pairs X_i, X_j under the conditions in Theorem B.1. □

Theorem B.3. *Consider an edge $X_i \to X_j$ in the true causal graph. The parameter γ_{ij} converges to $\sigma(\gamma_{ij}) = 1$ if the following condition holds:*

$$\min_{\hat{pa}\subseteq gpa_i(X_j)} \mathbb{E}_{\hat{I}\sim p_I(I)}\mathbb{E}_{\tilde{p}_{\hat{I}}(\boldsymbol{X})}\left[\log p(X_j|\hat{pa}, X_i) - \log p(X_j|\hat{pa})\right] > \lambda_{sparse} \tag{28}$$

where $gpa_i(X_j)$ is the set of nodes excluding X_i which, according to the ground truth graph, could have an edge to X_j without introducing a cycle.

Proof. To show this convergence, we assume that the orientation parameters have converged corresponding to Theorem B.1 and B.2. The parameter γ_{ij} converges to $\sigma(\gamma_{ij}) = 1$ if its gradient, $\frac{\partial}{\partial\gamma_{ij}}\tilde{\mathcal{L}}$, is negative independent of other values of $\boldsymbol{\gamma}$ and orientation parameters $\boldsymbol{\theta}$ that are not included in Theorem B.1 and B.2. The gradient of γ_{ij} includes an expectation over adjacency matrices $p_{\boldsymbol{\gamma},\boldsymbol{\theta}}(C)$. Based on the converged $\boldsymbol{\theta}$-values, we only need to consider sets of nodes as parents for X_j that contain parents, ancestors, or (conditionally) independent nodes according to the ground truth graph. This sets of parents is represented by $\text{gpa}_i(X_j)$. Among those remaining parent sets, we need to ensure that for any such set, the gradient is negative. The condition in Equation 28 corresponds to the inequality $\frac{\partial}{\partial\gamma_{ij}}\tilde{\mathcal{L}} < 0$ since the term on the left represents the log-likelihood difference $\mathcal{L}_{X_i\to X_j}(X_j) - \mathcal{L}_{X_i\not\to X_j}(X_j)$ in the gradients of γ_{ij} in Equation 20 with a flipped sign. For readability and better interpretation, λ_{sparse} has been moved on the right site of the inequality. This is possible as λ_{sparse} is independent of the two expectations in Equation 28. If the inequality holds for all parent sets $\hat{\text{pa}}$), the gradient of γ_{ij} can be guaranteed to be negative in expectation, independent of the other values of $\boldsymbol{\gamma}$. Since the distribution over parent sets $\hat{\text{pa}}$) depends on other values of $\boldsymbol{\gamma}$, the condition in Equation 28 ensures that even for the parent set with the lowest log-likelihood difference, it is still larger than λ_{sparse}. If this condition holds, then the gradient of $\frac{\partial}{\partial\gamma_{ij}}\tilde{\mathcal{L}}$ will be smaller than zero independent of other values of $\boldsymbol{\gamma}$. □

The condition in Equation 28 introduces a dependency between convergence guarantees and the regularizer parameter λ_{sparse}. The lower we set the regularization weight λ_{sparse}, the more edges we can guarantee to recover. If the regularization weight is set too high, we can eventually obtain false negative edge predictions. If the regularization weight is set very low, we take a longer time to converge as it requires lower gradient variance or more update steps, and is more sensitive in a limited data regime. Nonetheless, if sufficient computational resources and data is provided, any value of $\lambda_{\text{sparse}} > 0$ can be used.

Theorem B.4. *Assume for all edges $X_i \to X_j$ in the true causal graph, $\sigma(\theta_{ij})$ and $\sigma(\gamma_{ij})$ have converged to one. Then, the likelihood of all other edges,* i.e.$\sigma(\theta_{lk}) \cdot \sigma(\theta_{lk})$*, will converge to zero under the condition that $\lambda_{sparse} > 0$.*

Proof. If all edges in the ground truth graph have converged, all other pairs of variables X_l, X_k are (conditionally) independent in the graph. This statement follows from the Markov property of the graph and excludes ancestor-descendant pairs X_i, X_j. The possibility of having edges from descendants to ancestors has been removed by the fact that the orientation parameters θ_{ij} have converged according to Theorem B.2. Thus, for those cases, we already have the guarantee that $\sigma(\theta_{ij}) \cdot \sigma(\theta_{ij})$ converges to zero.

For a conditionally independent pair X_l, X_k, the difference of the log-likelihood estimate in the gradient of γ_{lk}, *i.e.*$\mathcal{L}_{X_l \to X_k}(X_k) - \mathcal{L}_{X_l \not\to X_k}(X_k)$, is zero in expectation since independent nodes do not share any information. Thus, the gradient remaining is:

$$\frac{\partial}{\partial \gamma_{lk}} \tilde{\mathcal{L}} = \sigma'(\gamma_{lk}) \cdot \sigma(\theta_{lk}) \cdot \lambda_{\text{sparse}} \tag{29}$$

Since the gradient is positive independent of the values of γ_{lk} and θ_{lk}, γ_{lk} will decrease until it converges to $\sigma(\gamma_{lk}) = 0$.

Hence, if γ_{lk} decreases for all pairs of (conditionally) independent variables X_l, X_k in the ground truth graph, and $\sigma(\theta_{lk})$ converged to zero for children and descendants, the product $\sigma(\gamma_{lk}) \cdot \sigma(\theta_{lk})$ will converge to zero for all edges not existing in the ground truth graph. □

For graphs that fulfill all conditions in the Theorems B.1 and B.4, ENCO is guaranteed to converge given sufficient data and time. The conditions in the theorems ensure that there exist no local minima or saddle points in the loss surface of the objective in Equation 2 with respect to $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$.

B.2.1 Graphs without guarantees

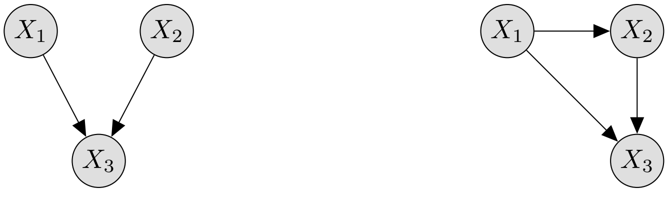
Most common causal graphs fulfill the conditions mentioned above as long as a small enough value for λ_{sparse} is chosen. Still, there are situations where we cannot guarantee that ENCO convergences to the correct causal graph independent of the chosen value of λ_{sparse}. Here, we want to discuss two scenarios visualized in Figure 6 under which the guarantees fail. Still, we want to emphasize that despite graphs not fulfilling the conditions, ENCO might still converge to the correct DAG for those as the guarantee conditions assume the worst-case scenarios for $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$ in all situations.

The first example we discuss is based on a fork structure where we have three binary variables, $\{X_1, X_2, X_3\}$, and the edges $X_1 \to X_3$ and $X_2 \to X_3$ (see Figure 6a). The parameterization we look at is a (noisy) XOR-gate for X_3 with its two input variables X_1, X_2 being independent of each other and uniformly distributed. The conditional probability distribution $p(X_3|X_1, X_2)$ can be summarized in the following probability table:

$p(X_3 = 1 \vert X_1, X_2)$	$X_2 = 0$	$X_2 = 1$
$X_1 = 0$	ϵ	$1 - \epsilon$
$X_1 = 1$	$1 - \epsilon$	ϵ

In other words, if $X_1 \neq X_2$, X_3 is equals 1 with a likelihood of $1 - \epsilon$. If $X_1 = X_2$, X_3 is equals 1 with a likelihood of ϵ. The issue that this probability table creates is the following. Knowing only one out of the two variables does not improve the log likelihood estimate for the output. This is because X_1 and X_2 are independent of each other, and $p(X_3|X_1) = p(X_3)$ is a uniform distribution. Hence, the worst-case parent set in Equation 6 would be the empty set, and leads to an expected difference log-likelihood difference of zero. As λ_{sparse} is required to be greater than zero for Theorem B.4, we cannot fulfill the condition for that graph. This means that an empty graph without any edges is a local minimum to which ENCO could converge. Yet, when the edge probabilities are non-zero, we will sample adjacency matrices with both input variables being a parent of X_3 with a non-zero probability. Hence, the log-likelihood difference for X_1 and X_2 to X_3 is unequal zero. Hence, this graph is still often correctly discovered despite ENCO not having a convergence guarantee for it.

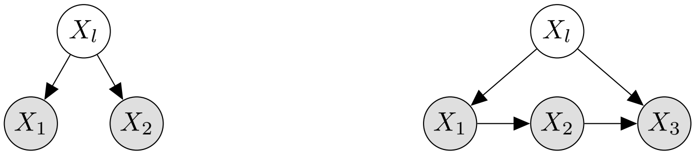
The second example we want to discuss aims at graphs that violate the condition in Theorem B.1, more specifically Equation 23. The graph we consider is a fully connected graph with three variables X_1, X_2, X_3 (see Figure 6b). The scenario can be described as follows: if knowing X_2 informs the log-likelihood estimate of X_3 more about X_1 than about X_2 itself, an intervention on X_2 and a sampled graph with the edge $X_2 \to X_3$ could lead to a worse likelihood estimate of X_3 than



(a) Fork example

(b) Fully connected example

Figure 6: Causal graph structures for which, under specific parameterization of the conditional distributions, the conditions for guaranteeing convergence can be violated.



(a) Independent children

(b) Ancestor-descendant pair

Figure 7: Visualization of the different graph structures we need to consider in the guarantee discussion of latent confounder detection. Latent variables X_l are shown in white, all other variables are observed. (a) The two children of X_l, X_1 and X_2, are independent of each other. (b) X_1 is an ancestor of X_3, and the two variables have a shared latent confounder X_l.

without the edge. For this scenario to happen, $p(X_2|X_1)$ must be close to deterministic. Additionally, $p(X_3|X_1, X_2)$ must be much less reliant on X_2 than on X_1, such as in the following probability table:

$p(X_3 = 1\|X_1, X_2)$	$X_2 = 0$	$X_2 = 1$
$X_1 = 0$	ϵ_1	ϵ_2
$X_1 = 1$	$1 - \epsilon_1$	$1 - \epsilon_2$

The two variables ϵ_1, ϵ_2 represent small constants close to zero. In this case, the graph can violate the condition in Equation 23 since intervening on X_2 breaks the dependency between X_1 and X_2. The conditional distribution $p(X_3|X_2)$ learned from observational data relies on the dependency between X_1 and X_2 which can make it to a worse estimate than $p(X_3)$. This graph could also not be reliably detected by other methods such as SDI [18] and DCDI [4] since most continuous optimization methods rely on the idea that adding an edge from the true causal graph improves the log-likelihood estimate. Nonetheless, we did not observe any of these situations in the synthetic and real-world graphs we experimented on. Furthermore, when $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$ are not initialized with the worst-case values, the graph with both X_1 and X_2 as parents of X_3 can be sampled and provides gradients in the correct direction.

B.3 Convergence conditions for latent confounder detection

In Section 3.5, we have discussed that ENCO can be extended to graph with latent confounders. For this, we have to record the gradients of γ_{ij} for the interventional data on X_i and all other interventional data separately. We define $\gamma_{ij} = \gamma_{ij}^{(I)} + \gamma_{ij}^{(O)}$ where $\gamma_{ij}^{(I)}$ is only updated with gradients from Equation 3 under interventions on X_i, and $\gamma_{ij}^{(O)}$ on all others. The score to detect latent confounders is:

$$\text{lc}(X_i, X_j) = \sigma\left(\gamma_{ij}^{(O)}\right) \cdot \sigma\left(\gamma_{ji}^{(O)}\right) \cdot \left(1 - \sigma\left(\gamma_{ij}^{(I)}\right)\right) \cdot \left(1 - \sigma\left(\gamma_{ji}^{(I)}\right)\right) \tag{30}$$

In this section, we show under which conditions the score $\text{lc}(X_i, X_j)$ converges to one if X_i and X_j share a latent confounder. We restrict our discussion to latent confounders between two variables that do not have any direct edges with each other, and assume that the confounder is not a child of any other observed variable. We assume that the causal graph based on the observed variable fulfills all conditions of Theorem B.1 to B.4 in Section B.2, meaning that without the latent confounders, the graph could have been recovered without errors. Under those conditions, we can also show that the graph among observed variables with latent confounders is also correctly recovered. This is since the latent confounders only affect Theorem B.4: if X_i and X_j share a latent confounder, they are not conditionally independent given their observed parents. Thus, we can rely on the fact that all edges in the true causal graph will be found according to Theorem B.1 to B.4, and the edges with latent confounders do not fulfill Theorem B.4.

For all pairs of variables that do not share a latent confounder, $\text{lc}(X_i, X_j)$ converges to zero. The edges that are removed in Theorem B.4 converge to $\sigma(\gamma_{ij}^{(O)}) = 0$ which sets $\text{lc}(X_i, X_j)$ to zero. For edges that have been recovered, we state in Equation 24 that the gradient for interventional data must be negative for interventions on the parent. Hence, $\sigma(\gamma_{ji}^{(I)})$ converges to one which brings $\text{lc}(X_i, X_j)$ to zero again.

For variables that share a latent confounder, we distinguish between two cases that are visualized in Figure 7. In the first case, we assume that X_i and X_j are independent in the true causal graph excluding the latent confounder. This means that an intervention on X_i does not cause any change in X_j, and vice versa. The second case describes the situation where X_i is an ancestor of X_j. The case of X_i being a parent of X_j has been excluded in earlier assumptions as in those cases, we cannot separate the causal effect of X_i on X_j based on its causal relation and the latent confounder.

In case that the two children of the latent confounder are not an ancestor-descendant pair, we can provide a guarantee under the following conditions.

Theorem B.5. *Consider a pair of variables X_i, X_j that share a latent confounder X_l. Assume that X_i and X_j are conditionally independent given the latent confounder and their observed parents. Further, all other edges in the causal graph have been recovered under the conditions of Theorem B.1 to B.4. The confounder score $lc(X_i, X_j)$ converges to one if the following two conditions hold:*

$$\mathbb{E}_{\hat{I}\sim p_{I\text{-}X_i}(I)}\mathbb{E}_{\tilde{p}_{\hat{I}}(\boldsymbol{X})}\left[\log p(X_j|pa(X_j), X_i) - \log p(X_j|pa(X_j))\right] > \lambda_{sparse} \tag{31}$$

$$\mathbb{E}_{\hat{I}\sim p_{I\text{-}X_j}(I)}\mathbb{E}_{\tilde{p}_{\hat{I}}(\boldsymbol{X})}\left[\log p(X_i|pa(X_i), X_j) - \log p(X_i|pa(X_i))\right] > \lambda_{sparse} \tag{32}$$

Proof. We need to show that under the two conditions above, $\sigma(\gamma_{ij}^{(O)})$ and $\sigma(\gamma_{ji}^{(O)})$ are guaranteed to converge to one while $\sigma(\gamma_{ij}^{(I)})$ and $\sigma(\gamma_{ji}^{(I)})$ converge to zero. The distribution $p_{I\text{-}X_k}(I)$ represents the distribution over interventions excluding the ones performed on the variable X_k. The two conditions resemble Equation 28 with the difference that the intervention on the potential parent variable is excluded, and the parent set is the true parent set of the correct causal graph. This is because all other edges have been correctly recovered, and the two conditions are concerning $\sigma(\gamma_{ij}^{(O)})$. If the condition in Equation 31 holds, it corresponds to a negative gradient in $\gamma_{ij}^{(O)}$ following the argumentation in Theorem B.3. The same holds for $\gamma_{ji}^{(O)}$. Therefore, $\sigma(\gamma_{ij}^{(O)})$ and $\sigma(\gamma_{ji}^{(O)})$ are guaranteed to converge to one under the conditions given in Theorem B.5.

For the interventional parameters $\gamma_{ij}^{(I)}$ and $\gamma_{ji}^{(I)}$, we show that the gradient can only be positive, *i.e.*decreasing $\gamma_{ij}^{(I)}$ and $\gamma_{ji}^{(I)}$. Under the intervention on X_i, X_i and X_j become independent since we assume perfect interventions. In this case, the log-likelihood estimate of X_j cannot be improved by conditioning on X_i. Hence, the difference $\mathcal{L}_{X_i\to X_j}(X_j) - \mathcal{L}_{X_i\not\to X_j}(X_j)$ is greater or equal to zero. When further considering the sparsity regularizer λ_{sparse}, the gradient of γ_{ij} under interventions on X_i can only be positive, *i.e.*decreasing $\gamma_{ij}^{(I)}$. The same argument holds for $\gamma_{ji}^{(I)}$. Thus, we can conclude that $\sigma(\gamma_{ij}^{(I)})$ and $\sigma(\gamma_{ji}^{(I)})$ converge to zero. □

If the conditions of Theorem B.5 are not fulfilled, $\sigma(\gamma_{ij}^{(O)})$ and $\sigma(\gamma_{ji}^{(O)})$ might converge to zero. This results in the score $\text{lc}(X_i, X_j)$ being zero, but also $\sigma(\gamma_{ij})$ converging to zero. Hence, we do not get any false positive edge predictions as we have seen in the experiments of Section 4.4.

For the second case where X_i is an ancestor of X_j, we cannot give such a guarantee because of Theorem B.2. Theorem B.2 states that $\sigma(\theta_{ij})$ converges to one for ancestor-descendant pairs.

However, $\sigma(\theta_{ji})$ is a factor in the gradients of γ_{ji}. This means that if $\sigma(\theta_{ji})$ converges to zero according to Theorem B.2, we cannot guarantee that γ_{ji} converges to the desired value since its gradient becomes zero. Nevertheless, 59.2% of the latent confounders in our experiments of Section 4.4 were on ancestor-descendant pairs. ENCO detects a majority of those confounders, showing that ENCO still works on such confounders despite not having guarantees. Further, we show in Section C.3 that the confounder scores lc(X_i, X_j) indeed converge to one for detected confounders, and zero for all other edges.

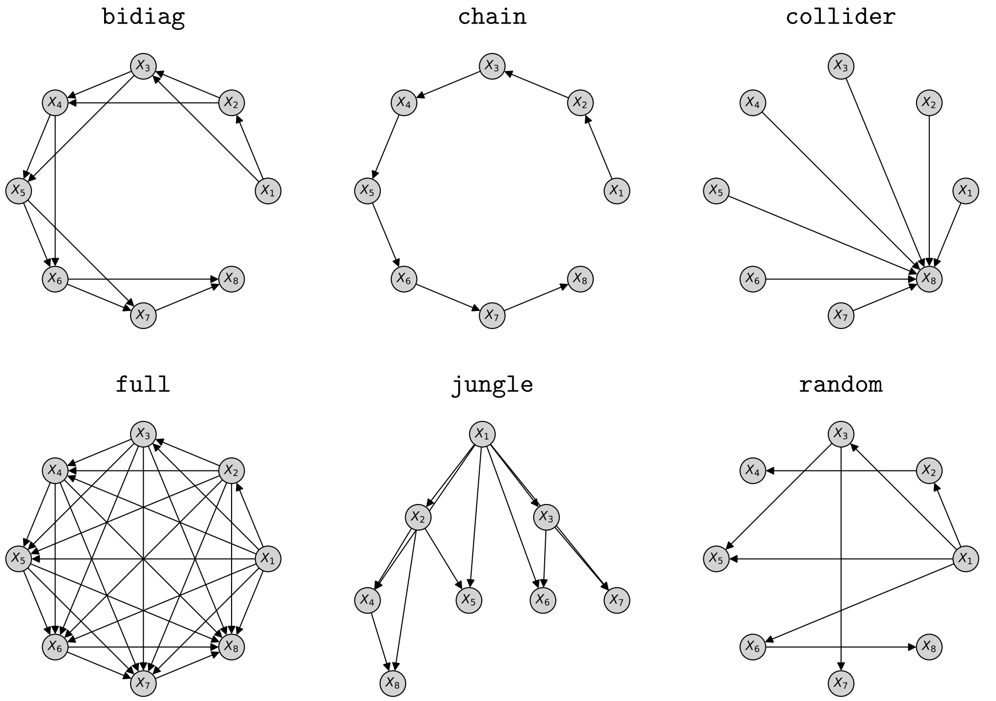


Figure 8: Visualization of the common graph structures for graphs with 8 nodes. The graphs used in the experiments had 25 nodes. Note that the graph `random` is more densely connected for larger graphs as the number of possible edges scales quadractically with the number of nodes.

C Experimental details

The following section gives an overview of the hyperparameters used across experiments. Additionally, we discuss details of the graph generation and the learning process of different algorithms.

C.1 Common graph structure experiments

C.1.1 Datasets

Graph generation The six common graph structures we have used for the experiments in Table 1 are visualized in Figure 8. In the graph `bidiag`, a variable X_i has X_{i-2} and X_{i-1} as parents, and consequently X_{i+1} and X_{i+2} as children. Hence, this graph represents a chain with 2-hop connections. The graph `chain` is a `bidiag` with a single hop, meaning that X_i is the parent of X_{i-1} but not X_{i-2}. In the graph `collider`, the variable X_N has all other variables, $\boldsymbol{X}_{-i}$, as parents. In the graph `full`, the parents of a variable X_i are all previous variables: $\text{pa}(X_i) = \{X_1, X_2, ..., X_{i-1}\}$. Hence, it is the densest connected graph possible. The graph `jungle` represents a binary tree where a node is also connected to its parent's parent. Finally, the graph `random` follows a randomly sampled adjacency matrix. For every possible pair of variables X_i, X_j, we sample an edge with a likelihood of 0.3. To determine the orientation of the edges, we assume the causal ordering of $X_i \succ X_{i+1}$. In other words, if we have an edge between X_i and X_j, it is oriented $X_i \rightarrow X_j$ is $i < j$ else $X_j \rightarrow X_i$.

Conditional distributions In the generated graphs, we use categorical variables that each have 10 categories. To model the ground-truth conditional distributions, we use randomly initialized neural networks. Specifically, we use MLPs of two layers where the categorical inputs are represented by embedding vectors. For a variable X_i with M parents, we stack the M embedding vectors to form the input to the following MLPs. Each embedding has a dimensionality of 4, hence the input size to the first linear layer is $4M$. The hidden size of the layers is 48, and we use a LeakyReLU activation function in between the two linear layers. Finally, a softmax activation function is used on the output to obtain a distribution over the 10 categories. The MLP and hyperparameters have been chosen

Table 4: Hyperparameter overview for the simulated graphs dataset experiments presented in Table 1.

Hyperparameters	SDI	ENCO
Sparsity regularizer λ_{sparse}	$\{0.01, \underline{0.02}, 0.05, 0.1, 0.2\}$	$\{0.002, \underline{0.004}, 0.01\}$
DAG regularizer	$\{0.2, \underline{0.5}, 1.0, 2.0, 5.0\}$	-
Distribution model	2 layers, hidden size 64, LeakyReLU($\alpha = 0.1$)	
Batch size	128	
Learning rate - model	$\{$2e-3, $\underline{\text{5e-3}}$, 2e-2, 5e-2$\}$	
Distribution fitting iterations F	1000	
Graph fitting iterations G	100	
Graph samples K	100	
Epochs	50	30
Learning rate - γ	$\{\underline{\text{5e-3}}$, 2e-2, 5e-2$\}$	$\{$5e-3, $\underline{\text{2e-2}}$, 5e-2$\}$
Learning rate - $\boldsymbol{\theta}$	-	$\{$5e-3, 2e-2, 5e-2, $\underline{\text{1e-1}}\}$

based on the design of networks used in ENCO, SDI [18] and DCDI [4]. For the initialization of the networks, we follow Ke et al. [18] and use the orthogonal initialize with a gain of 2.5. The biases are thereby initialized uniformly between -0.5 and 0.5. This way, we obtain non-trivial, random distributions.

C.1.2 Methods and hyperparameters

Baseline implementation We used existing implementations to run the baselines GIES [13], IGSP [47], and DCDI [4]. For GIES, we used the implementation from the R package `pcalg`[2]. To run categorical data, we used the `GaussL0penIntScore` score function. For IGSP, we used the implementation of the python package `causaldag`[3]. As IGSP uses conditional independence tests in its score function, we cast the categorical data into continuous space first and experiment with different kernel-based independence tests. Due to its long runtime for large dataset sizes, we limit the interventional and observational data set size to 25k. Larger dataset sizes did not show any significant improvements. Finally, we have used the original python code for DCDI published by the authors[4]. We have added the same neural networks used by ENCO into the framework to perform structure learning on categorical data. Bugs in the original code were corrected to the best of our knowledge. Since SDI [18] has a similar learning structure as ENCO, we have implemented it in the same code base as ENCO. This allows us to compare the learning algorithms under exact same perquisites. Further, all methods with neural networks used the deep learning framework PyTorch [29] which ensures a fair run time comparison across methods.

Hyperparameters To ensure a fair comparison, we performed a hyperparameter search for all methods. The hyperparameter search was performed on a hold-out set of graphs containing two of each graph structure.

GIES We performed a hyperparameter search over the regularizer values $\lambda \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200\}$. The value obtaining the best results in terms of structural hamming distance (SHD) was $\lambda = 20$. The average run time of GIES was 2mins per graph.

IGSP We experimented with two different conditional independence tests, `kci` and `hsic`, and different cutoff values $\alpha = \{\text{1e-5}, \text{1e-4}, \text{1e-3}, \text{1e-2}\}$. The best hyperparameter setting was `kci` with $\alpha =$ 1e-3. The average run time of IGSP was 13mins.

SDI We focused the hyperparameter search for SDI on its two regularizers, λ_{sparse} and λ_{DAG}, as well as its learning rate for γ. The other hyperparameters with respect to the neural networks were kept the same as ENCO for a fair comparison. We show all details of the hyperparameter search in Table 4. The best combination of regularizers found was $\lambda_{\text{sparse}} = 0.02$ and

[2] `https://cran.r-project.org/web/packages/pcalg/index.html`

[3] `https://github.com/uhlerlab/causaldag`

[4] `https://github.com/slachapelle/dcdi`

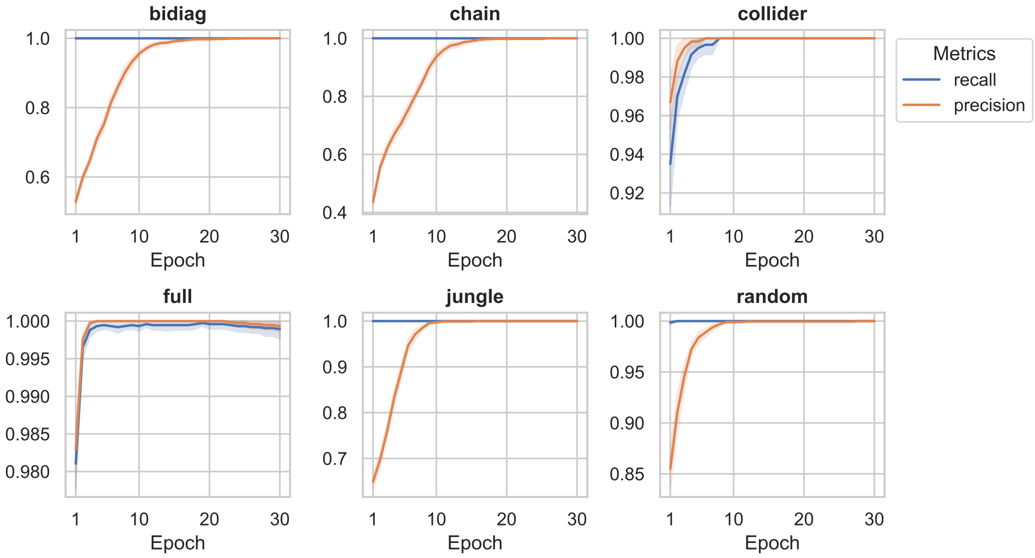


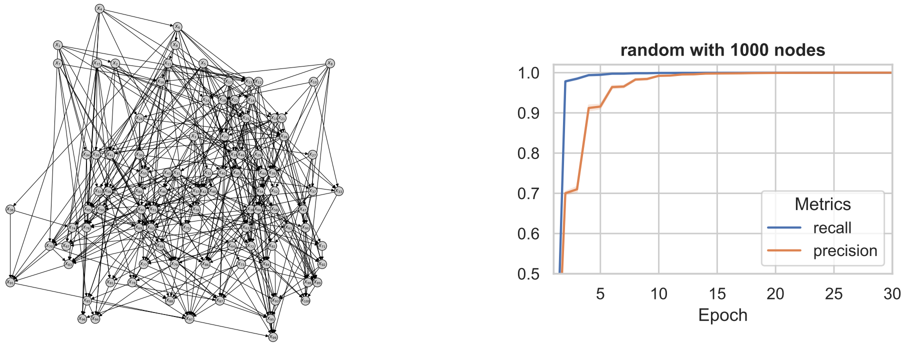
Figure 9: Learning curves of ENCO in terms of recall and precision on edge predictions for synthetic graph structures. The orientations for the ground-truth edges are not plotted as they have usually been correctly learned after the first epoch except for the graph `full`. Overall, we see that the edge recall starts very high for all graphs, and the precision catches up over the epochs. This is in line the steps in the convergence proof in Section B.

$\lambda_{\text{DAG}} = 0.5$. Lower values of λ_{sparse} lead to more false positives, especially in sparse graphs, while a lower value of λ_{DAG} caused many two-variable loops. Compared to the reported hyperparameter by Ke et al. [18] ($\lambda_{\text{DAG}} = 0.5, \lambda_{\text{sparse}} = 0.1$), we found a lower sparsity regularizer to work better. This is likely because of testing SDI on larger graphs. In contrast to ENCO, SDI needed a lower learning rate for γ due to its higher variance gradient estimators. To compensate for it, we ran it for 50 instead of 30 epochs. The average run time of SDI was 4mins per graph.

DCDI The most crucial hyperparameter in DCDI was the initialization of the constraint factor μ_0. We experimented with a range of $\mu_0 \in \{1\text{e-}10, 1\text{e-}9, 1\text{e-}8, 1\text{e-}7, 1\text{e-}6, 1\text{e-}5\}$ and found $\mu_0 = 1\text{e-}9$ to work best. This is close to the reported value of 1e-8 by Brouillard et al. [4]. Higher values lead to empty graphs, while lower values slowed down the optimization. Additionally, we search over the regularizer hyperparameter $\lambda \in \{1\text{e-}3, 1\text{e-}2, 1\text{e-}1, 1\text{e-}0, 1\text{e}1\}$ where we found $\lambda = 0.1$, which is the same value used by Brouillard et al. [4]. We stop the search after the Lagrangian constraint is below 1e-7, following Brouillard et al. [4], or 50k iterations have been used which was sufficient to converge on all graphs. The average run time of DCDI was 16 minutes.

ENCO We outline the hyperparameters of ENCO in Table 4. As discussed in Section 3.4, the most crucial hyperparameter in ENCO is the sparsity regularizer λ_{sparse}. The larger it is, the faster ENCO converges, but at the same time might miss edges in the ground-truth graph. Lower values allow the detection of more edges for the price of longer training times. We have found that for the graph structures given, only the graph `full` was sensitive to the value of λ_{sparse} where $\lambda_{\text{sparse}} = 0.002$ and $\lambda_{\text{sparse}} = 0.004$ performed almost equally well. In comparison to SDI, ENCO can make use of larger learning rates due to lower variance gradient estimators. Especially for $\boldsymbol{\theta}$, we have noticed that high learning rates are beneficial. This is in line with our theoretical guarantees which require the orientation parameters to converge first. We use the Adam optimizer for $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$ with the β-hyperparameters $(0.9, 0.9)$ and $(0.9, 0.999)$ respectively. A lower β_2 hyperparameter for $\boldsymbol{\gamma}$ allows the second momentum to adapt faster to a change of gradient scale which happens for initial false positive predictions.

The average run time of ENCO was 2mins per graph. The algorithm could be sped up even more by reducing the number of graph samples K and model fitting iterations. However,



(a) Example graph 100 nodes

(b) Learning curve $1,000$ nodes

Figure 10: (a) Example graph of 100 variables (best viewed electronically). Every node has on average 8 edges and a maximum of 10 parents. (b) Plotting recall and precision of the edge predictions for the training on graphs with $1,000$ nodes. The small standard deviation across graphs shows that ENCO can reliably recover large graphs.

for graphs of larger than 100 nodes, $K = 100$ and longer model fitting times showed to be beneficial. The learning curves in terms of recall and precision are shown in Figure 8.

C.2 Scalability experiments

Graph generation For generating the graphs, we use the same strategy as for the graph `random` in the previous experiments. The probability of sampling an edge is set to $8/N$, meaning that on average, every node has 8 in- and outgoing edges. We limit the number of parents to 10 per node since otherwise, we cannot guarantee that the randomly sampled distributions take all parents faithfully into account. This is also in line with the real-world inspired graphs of the BnLearn repository, which have a maximum of 6 parents. To give an intuition on the complexity of such graphs, we show an example graph of 100 nodes in Figure 10a. Since the datasets for such graphs can get very large, we limit the interventional datasets to $4,096$ instead of $10,000$ data points.

γ-freezing stage For ENCO, one challenge of large graphs is that the orientation parameters $\boldsymbol{\theta}$ are updated very sparsely. The gradients for θ_{ij} require data from an intervention on one of its adjacent nodes X_i or X_j, which we evaluate less frequently with increasing N as we iterate over interventions on all N nodes. Hence, we require more iterations/epochs just for training the orientation parameters while wasting a lot of computational resources. To accelerate training of large graphs, we freeze $\boldsymbol{\gamma}$ in every second graph fitting stage. Updating only $\boldsymbol{\theta}$ allows us to use the same graph sample C_{-ij} for both $\mathcal{L}_{X_i \to X_j}(X_j)$ and $\mathcal{L}_{X_i \not\to X_j}(X_j)$ since the log-likelihood estimate of X_j only needs to be evaluated for θ_{ij}. With this gradient estimator, we experience that as little as 4 graph samples are sufficient to obtain a reasonable gradient variance. Hence, it is possible to perform more gradient updates of $\boldsymbol{\theta}$ in the same computation time. Note that this is estimator not efficient when training $\boldsymbol{\gamma}$ as we require different C_{-ij} samples for every i. In experiments, we alternate the standard graph fitting step with this pure θ-training stage. We want to emphasize that this approach can also be used for small graphs obtaining similar results as in Table 1. However, it is less needed because the orientation parameters are more frequently updated in the first place. Such an approach is not possible for the baselines, SDI and DCDI, because they do not model the orientation as a separate variable.

Hyperparameters To experiment with large graphs, we mostly keep to the same hyperparameters as reported in Section C.1. However, all methods showed to gain by a small hyperparameter search. For SDI and ENCO, we increase the number of distribution fitting iterations as the neural networks need to model a larger set of possible parents. We also increase the learning rate of $\boldsymbol{\gamma}$ to 2e-2. However, while SDI reaches better performance with the increased learning rate at epoch 30, it showed to perform worse when training for longer. This indicates that high learning rates can lead to local minima in SDI. Additionally, we noticed that a slightly higher sparsity regularizer improved convergence speed for ENCO while SDI did not improve with a higher sparsity regularizer. Table 5 shows a hyperparameter overview of ENCO on large-scale graphs, and Figure 10b the learning curve on graphs of $1,000$ nodes.

For DCDI, we noticed that the hyperparameters around the Lagrangian constraint needed to be carefully fine-tuned. The Lagrangian constraint can reach values larger than possible to represent with double, and starts with 8e216 for graphs of $1,000$ nodes. Following Brouillard et al. [4], we normalize the constraint by the value after initialization, which gives us a more reasonable value to start learning. We performed another hyperparameter search on μ_0, noticed however that it did not have a major impact. In the run time of ENCO, DCDI just starts to increase the weighting factor of the augmented Lagrangian while the DAG constraint is lower than 1e-10 for the smallest graph. The best value found was $\mu_0 = $ 1e-7.

Table 5: Hyperparameter overview of ENCO for the scalability experiments presented in Table 6. Numbers in the center represent that we use the same value for all graphs.

Hyperparameters	100 nodes	200 nodes	400 nodes	1000 nodes
Distribution model	2 layers, hidden size 64, LeakyReLU($\alpha = 0.1$)			
Batch size - model	128	128	128	64
Learning rate - model	5e-3			
Distribution fitting iterations F	2000	2000	4000	4000
Graph fitting iterations G	100			
Graph samples	100			
Learning rate - $\boldsymbol{\gamma}$	2e-2			
Learning rate - $\boldsymbol{\theta}$	1e-1			
θ-training iterations	1000	1000	2000	2000
θ-training graph samples	2 + 2			
Sparsity regularizer λ_{sparse}	{0.002, 0.004, $\underline{0.01}$}			
Number of epochs (max.)	30			

Results For clarity, we report the results of all methods below. The exact values are not easily readable in Figure 3 due to large differences in performance.

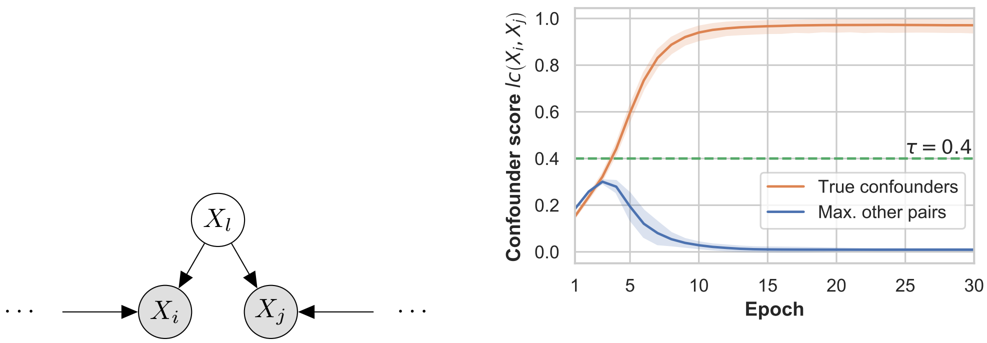
Table 6: Results for graphs between 100 and 1000 nodes. We report the average and standard deviation of the structural hamming distance (SHD) over 10 randomly sampled graphs. †The maximum runtime of ENCO was measured on an NVIDIA RTX3090. Baselines were executed on the same hardware.

Graph size **Max Runtime**†	100 15mins	200 45mins	400 2.5hrs	1000 9hrs
DCDI [4]	583.1 (±21.8)	1399.0 (±67.5)	4761.2 (±303.4)	OOM
SDI [18]	35.7 (±2.9)	100.9 (±7.6)	356.3 (±11.5)	1314.4 (±58.5)
ENCO	0.0 (±0.0)	0.0 (±0.0)	0.0 (±0.0)	0.2 (±0.42)

C.3 Latent confounder experiments

Graph generation The graphs used for testing the latent confounding strategy are based on the `random` graphs from Section 4.2. We use graphs of 25 nodes, and add 5 extra nodes that represent latent confounders. Each latent confounder X_l is connected to two randomly sampled nodes X_i, X_j that do not have a direct connection. However, X_i and X_j can be an ancestor-descendant pair and have any other (shared) parent set (see Figure 11a). In the adjacency matrix, we add the edges $X_l \rightarrow X_i$ and $X_l \rightarrow X_j$, and perform the data generation as for the previous graphs. After data generation, we remove the 5 latent confounders from both observational and interventional data. The task is to learn the graph structure of the remaining 25 observable variables, as well as detecting whether there exists a latent confounder between any pair of variables.

Hyperparameters To show that we can perform latent confounder detection without specific hyperparameters, we use the same hyperparameters as for the experiment on the previous graph structures (see Appendix C.3). To record $\gamma_{ij}^{(I)}$ and $\gamma_{ij}^{(O)}$ separately, we use separate first and second order momentum parameters in the Adam optimizer. We plot in Figure 11b the latent confounder scores



(a) Latent confounder example

(b) Threshold sensitivity

Figure 11: **Left**: Example of a latent confounder scenario, where X_l is not observed and introduces a dependency between X_i and X_j on observational data. The dots on the left and right represent eventual (observed) parents of X_i and X_j. **Right**: Plotting the average score $\text{lc}(X_i, X_j)$ for confounders $X_i \leftarrow X_l \rightarrow X_j$ in the true causal graph (orange) and maximum score of any other node pair (blue). The plot shows the detection of latent confounders in ENCO is not sensitive to the specific value of τ.

$\text{lc}(X_i, X_j)$ calculated based on Equation 8. We see that the score converges close to 1 for pairs with a latent confounder, and for all other, it converges to 0. This verifies our motivation of the score function discussed in Section 3.5, and also shows that the method is not sensitive to the threshold hyperparameter τ. We choose $\tau = 0.4$ which was slightly higher than the highest value recorded for any other pair at early stages of training.

C.4 Real-world inspired experiments

Datasets We perform experiments on a collection of causal graphs from the Bayesian Network Repository (BnLearn) [40]. The repository contains graphs inspired by real-world applications that are used as benchmarks in literature. We chose the graphs to reflect a variety of sizes and different challenges (rare events, deterministic variables, etc.). The graphs have been downloaded from the BnLearn website[5].

Hyperparameters We reuse most of the hyperparameters of the previous experiments. For all graphs less than 100 nodes, we use the hyperparameters of Appendix C.1, *i.e.* the synthetic graphs of 25 nodes. For all graphs larger than 100 nodes, we use the hyperparameters of Appendix C.2, *i.e.* the large-scale graphs. One exception is that we allow the fine-tuning of the regularizer parameter for both sets. For ENCO, we used a slightly smaller regularizer, $\lambda_{\text{sparse}} = 0.002$, for the small graphs, and a larger one, $\lambda_{\text{sparse}} = 0.02$, for the large graphs. Due to the large amount of deterministic variables, ENCO tends to predict more false positives in the beginning before removing them one by one. For SDI, we also found a smaller regularizer, $\lambda_{\text{sparse}} = 0.01$, to work best for the small graphs. However, in line with the results of Ke et al. [18], SDI was not able to detect all edges. Even lower regularizers showed to perform considerably worse on the child dataset while minor improvements were made on the small graphs. Hence, we settled for $\lambda_{\text{sparse}} = 0.01$. In terms of run time, both methods used 100 epochs for the small graphs and 50 for the large graphs.

Results The results including standard deviations can be found in Table 7. The low standard deviation for ENCO shows that the approach is stable across seeds even for large graphs. SDI has a zero standard deviation for a few graphs. In those cases, SDI converged to the same graph across seeds, but not necessarily the correct graph.

C.5 Additional experiments

In this section, we show additional experiments performed as ablation studies of ENCO. First, we discuss the effect of using the gradient estimators proposed in Section 3.4 compared to Bengio et al. [3]. Next, we show experiments on synthetic graphs with deterministic variables.

[5] `https://www.bnlearn.com/bnrepository/`

Table 7: Results on graphs from the BnLearn library measured in structural hamming distance (lower is better). Results are averaged over 5 seeds with standard deviations.

Dataset	cancer [20] (5 nodes)	earthquake [20] (5 nodes)	asia [21] (8 nodes)	sachs [37] (11 nodes)	child [41] (20 nodes)	alarm [2] (37 nodes)	diabetes [1] (413 nodes)	pigs [40] (441 nodes)
SDI	3.0 (±0.0)	0.4 (±0.5)	4.0 (±0.0)	7.0 (±0.0)	11.8 (±0.4)	24.6 (±1.5)	422.4 (±8.7)	18.0 (±1.6)
ENCO	**0.0** (±0.0)	**0.0** (±0.0)	**0.0** (±0.0)	**0.0** (±0.0)	**0.0** (±0.0)	**1.0** (±0.0)	**2.0** (±0.0)	**0.0** (±0.0)

Table 8: Extension of Table 1 with ablation study of using Bengio et al. [3] gradients with ENCO.

Graph type	bidiag	chain	collider	full	jungle	random
GIES [13]	47.4 (±5.2)	22.3 (±3.5)	13.3 (±3.0)	152.7 (±12.0)	53.9 (±8.9)	86.1 (±12.0)
IGSP [47]	33.0 (±4.2)	12.0 (±1.9)	23.4 (±2.2)	264.6 (±7.4)	38.6 (±5.7)	76.3 (±7.7)
SDI [18]	2.1 (±1.5)	0.8 (±0.9)	14.7 (±4.0)	121.6 (±18.4)	1.8 (±1.6)	1.8 (±1.9)
DCDI [4]	3.7 (±1.5)	4.0 (±1.3)	**0.0** (±0.0)	2.8 (±2.1)	1.2 (±1.5)	2.2 (±1.5)
ENCO (Ours)						
- Bengio et al. [3] grads	0.1 (±0.4)	**0.0** (±0.0)	**0.0** (±0.0)	1.9 (±1.5)	**0.0** (±0.0)	**0.0** (±0.0)
- Our gradient estimator	**0.0** (±0.0)	**0.0** (±0.0)	**0.0** (±0.0)	**0.3** (±0.9)	**0.0** (±0.0)	**0.0** (±0.0)

Table 9: Experiments on graphs with deterministic variables. The performance over 10 experiments is reported in terms of SHD with standard deviation in brackets. Despite not having guarantees, ENCO can recover most of the graphs with less than two errors.

Graph type	deterministic
SDI [18]	20.6 (±3.8)
ENCO (Ours)	**1.4** (±1.3)

C.5.1 Ablation study on gradient estimators

To analyze the importance of the low-variance gradient estimators in ENCO, we repeat the experiments on synthetic graph structure from Section 4.2 where the gradient estimators of ENCO have been replaced by those from Bengio et al. [3]. The results are shown in Table 8. Overall, the scores are very similar with minor differences for the graphs `full` and `bidiag`. In comparisons to the learning curves in Figure 9, the curves with the gradient estimator of Bengio et al. [3] are more noisy, with recall and precision jumping up and down. While ENCO easily converged early to the correct graphs for all graph types, this model often required the full 30 iterations to reach the optimal recovery.

The difference between the two gradient estimators becomes more apparent on large graphs. We repeated the experiments of Section 4.3 on the graphs with 100 nodes using the gradient estimator of Bengio et al. [3]. Within the 30 epochs, the model obtained an SHD of 15.4 on average over 5 experiments, which is considerably higher than ENCO with the proposed gradient estimators (0.0). Still, this is only half of the errors that SDI [18] with the same gradient estimator achieved. Hence, we can conclude that the proposed gradient estimators are beneficial for ENCO but not strictly necessary for small graphs. For large graphs, the low variance of the estimator becomes much more important.

C.5.2 Deterministic variables

In contrast to algorithms working on observational data, ENCO does not strictly require the faithfulness assumption. Hence, we can apply ENCO to graphs with deterministic variables. Deterministic variables have a distribution that is defined by a one-to-one mapping of its parents' inputs to an output value. In other words, we have the following distribution:

$$p(X_i|\text{pa}(X_i)) = \mathbb{1}\left[X_i = f(\text{pa}(X_i))\right] \tag{33}$$

where f is an arbitrary function. The difficulty of deterministic variables is that a variable X_i can be fully replaced by its parents $\text{pa}(X_i)$ in any conditional distribution. The only way we can identify deterministic variables is from interventional data where an intervention on X_i breaks the dependency to its parents.

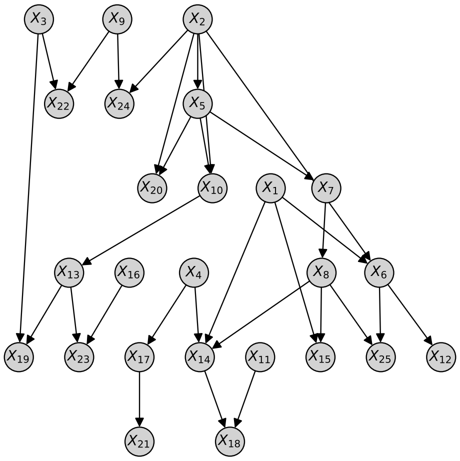


Figure 12: Example graph for the deterministic setup. We use the `random` setup with edge probability 0.1 and limit number of parents to 2. All variables except the leaf nodes have deterministic distributions.

We have already tested ENCO on deterministic variables in the context of the real-world inspired graphs of Section 4.5. To have a more detailed analysis, we created synthetic graphs following the `random` graph setup with an edge probability of 0.1 and a maximum of two parents. An example graph is shown in Figure 12. All variables except the leaf nodes have deterministic distributions, where the function $f(\text{pa}(X_i))$ is randomly created by picking a random output category for any pair of input values. We create 10 such graphs and use the same hyperparameter setup as for the synthetic graphs, except that we increase the sparsity regularizer to $\lambda_{\text{sparse}} = 0.02$. We report the results in Table 9. In line with the results on the real-world graphs, ENCO is able to recover most graphs with less than two errors. The error rate is only slightly higher than the `full` graph in the previous synthetic experiments. Note that guarantees cannot be given because the observational data might not cover all input value combinations. This means that the neural networks trained to model the conditional distributions cannot model all parent sets correctly. Nonetheless, the low discrepancy of the predicted graphs to the true causal graphs shows that a lot of relations can still be discovered. As a baseline, we apply SDI [18] and see a significant higher error rate. The method predicts many false positives including two-variable loops but was also missing out some true positive edges. We conclude that ENCO also works well on deterministic graphs, despite not strictly providing a guarantee for them.