IT UNIVERSITY OF COPENHAGEN

Learning Neural Causal Models With Adversarial Training

STADS: KISPECI1SE

August Wester
IT University of Copenhagen
auwe@itu.dk

June 2022

Contents

Abstract

In this paper, we motivate the need for incorporating causal models into machine learning systems. We cover fundamental concepts of causal inference such as structural causal models, interventions, and d-separation. We introduce the problem of causal discovery, i.e. the problem of inferring causal relationships from data, with an emphasis on the recent field of neural causal discovery. We briefly introduce the problem of generative modeling, with a particular focus on generative adversarial networks. Finally, we propose a novel method of score-based causal discovery using interventional data, combining ideas in the literature on neural causal discovery with the training modality of generative adversarial networks. We evaluate our proposed method on synthetic datasets of up to 10 variables, and we analyze its sensitivity to regularization. [1]

1 Introduction

The preceding decade has witnessed a surge of interest in data science and machine learning, in no small part due to advances in the field of deep learning. Characterized by the application of deep neural networks running on high-powered GPUs, breakthroughs in deep learning have made feasible a large swath of high-dimensional learning tasks that were previously beyond the reach of more traditional machine learning methods. Problems such as image classification (Krizhevsky et al. 2012; Szegedy et al. 2017), natural language processing (Vaswani et al. 2017; Brown et al. 2020), speech recognition (Hinton et al. 2012; He et al. 2019), and game-playing (Mnih et al. 2015; Silver, Schrittwieser, et al. 2017; Silver, Hubert, et al. 2017) have already seen significant advances, with more problems routinely succumbing to the power of large neural networks (Jumper et al. 2021; Ramesh et al. 2022).

However, despite its undeniable successes, some critics are drawing attention to the fact that current methods remain fundamentally limited in some respects (Marcus et al. 2019; Hartnett 2018a; Pearl 2019). Among the most common critiques is that current deep learning models exhibit poor generalization to **out-of-distribution** data. While many efforts have been made to alleviate this problem (e.g. by the use of data augmentation (Shorten et al. 2019) or outlier detection (Du et al. 2022)), some are arguing that generalizing well outside the i.i.d. setting requires a learner to not only model statistical correlations but rather to model **causal relationships** (Schölkopf et al. 2021). According to this view, improving a system's ability to generalize beyond its training distribution requires it not to model the data but to model the *mechanisms generating the data*. Interestingly, this view has parallels to that of some philosophers who, following Karl Popper in his rejection of inductivism (i.e. the idea that knowledge consists in extrapolating previous experiences to future ones), instead advocate that knowledge is created by conjecturing and criticizing *explanations* (Popper 2014; Deutsch 2012; Deutsch 2011). In this light, one can view a causal model as a rudimentary explanation; a representation of *how the data came to be*. As we will see, such representations allow a model to reason about the effect of interventions and, in some cases, even allow it to answer counterfactual questions (i.e. questions about hypothetical scenarios). These capabilities are at the core of human cognition (Gopnik et al. 2004; Sloman et al. 2009; Khemlani et al. 2014), making it likely that some form of causal modeling will play a role in the development human-level artificial intelligence. Despite this, the fields of machine learning and causality have developed largely independently, with cross-pollination between the two happening only in recent years. This cross-pollination is in large part thanks to the pioneering work of Pearl 2009, who has long argued for the importance of incorporating causal models into machine learning systems (Hartnett 2018b; Pearl 2019).

While the field of causality has largely concerned itself with the inferences one can make when *given* a causal model (often produced by human experts and over a relatively small number of variables), the associated field of **causal discovery** (also known as **structure learning**) has aimed at developing methods for inferring the causal structure underlying a given set of data (Spirtes et al. 2000). This problem is of particular interest in machine learning where the aspiration is, by and large, for models to uncover the causal structure of the world tabula rasa rather than by the use of handcrafted inductive biases.

[1] Code available at: `https://github.itu.dk/auwe/adversarial-ncm`

Following the success of deep learning, researchers are now considering the question of how to represent and embed causal models within larger neural network-based frameworks (Schölkopf et al. 2021). The confluence of these problems has given rise to recent research on neural causal discovery (Bengio et al. 2019; Ke et al. 2020; Zhu et al. 2019). The longer term ambition of this line of work is to address the open problem of how to uncover causal structure on the basis of high-dimensional sensory data, thus combining the strengths of deep learning with the predictive and explanatory power of causal models. This is a central aspect of the nascent field of **causal representation learning** (Schölkopf et al. 2021). However, for reasons that will be made clear later in this paper, most causal discovery techniques continue to operate in relatively low-dimensional domains in which the variables of interest are given a priori, while the causal relationships between them are either partially or completely obscured. Despite this, immediate applications of these methods abound, particularly within scientific disciplines such as biology (Friedman et al. 2000), epidemiology (Robins et al. 2000), the social sciences (Pearl 2009), and economics (Pearl 2009; Hicks et al. 1980). For instance, advances in gene editing technologies have in recent years provided ample amounts of interventional data from which causal structure can often be inferred (Jinek et al. 2012; Dixit et al. 2016). These efforts can help researchers deepen their understanding of the causal structure underlying complex natural phenomena and, consequently, provide meaningful insights into the effect of interventions.

In this work, we follow in the footsteps of recent research in the nascent field of neural causal discovery (Kocaoglu et al. 2017; Ke et al. 2020; Brouillard et al. 2020; Scherrer et al. 2021). We explore a novel training procedure based on the generative adversarial framework proposed by Goodfellow et al. (2014). We show how this training procedure can simultaneously be used as a method of training a generative model known as a neural causal model (Xia et al. 2021) while also providing a means of causal discovery in the form of a **probabilistic or soft adjacency matrix**. We evaluate our proposed method on datasets drawn from linear and nonlinear causal models of up to $N = 10$ variables. Finally, we offer some thoughts on future directions of research as well as a conclusion based on our findings.

2 Motivation

To further motivate the need for causal representation learning, we borrow the following illustrative example from Peters, Janzing, et al. (2017): Consider a simple scenario in which a learner is tasked with modeling the joint distribution of two variables: average altitude (A) and temperature (T) of cities in a given country. Given our prior knowledge of physics, we know that the observed correlation between A and T is likely due to a causal effect of altitude on temperature, expressed by the conditional distribution $P(T \mid A)$. The joint distribution, however, allows for two factorizations:

$$P(A, T) = P(A)P(T \mid A) \tag{1}$$

$$P(A, T) = P(T)P(A \mid T) \tag{2}$$

Here, the factorization in (1) obeys the causal direction while the factorization in (2) does not. In the i.i.d. setting, this distinction is inconsequential; each distribution works equally well assuming no distribution shift. It is only when we violate the i.i.d. assumption and change the joint distribution (e.g. by applying the same model in a different country) that differences start to emerge. Given that the conditional distribution $P(T \mid A)$ models the **mechanism** generating temperature from altitude, we can expect it to remain largely invariant across countries (disregarding exogenous factors like seasonal changes and distance to the equator), while the conditional $P(A \mid T)$ will exhibit no such robustness. Schölkopf et al. (2021) thus refer to (1) as the **causally disentangled factorization**. In the case just described, the benefit of the causally disentangled factorization becomes apparent when $P(A, T)$ changes but the model only needs to update $P(A)$. The same cannot be said of the **causally entangled factorization** in (2) which will need to update both $P(T)$ and $P(A \mid T)$. The utility of this fact was cemented by Bengio et al. (2019) when they showed how correctly assuming the causal structure in a bivariate setting leads to faster convergence when adapting to a transfer distribution. Interestingly, the authors demonstrated how the speed of adaptation can serve as a signal for a model to infer the correct causal structure, thereby providing a novel method of causal discovery. This, however, was only shown in the bivariate setting.

While the previous scenario is a toy example, it illustrates well how taking a causal approach to data modeling can deliver benefits that would not be possible otherwise. More generally, the notion of causally disentangled factorizations is closely related to what Schölkopf et al. (2021) refer to as **sparse mechanism shift**. This hypothesizes that changes in distribution tend to manifest themselves in a sparse or local way in the causally disentangled factorization. In other words, potentially large distribution shifts are often the result of changes in a small number of causal factors (e.g. putting on sunglasses produces a significant distributional shift in one's visual field but in the right representational

space, that change is described by only a single bit).

A perhaps even more important aspect of causal models is their ability to predict the effect of interventions; something which can *only* be done in the presence of a causal model. Interestingly, the theory of causal inference shows that this capacity is unattainable given observational data only (i.e. in the absence of experimental data). To see this, consider again the problem of modeling altitudes A and temperatures T in a given region. Absent any prior causal knowledge, observing the joint distribution $P(A, T)$ provides no insight into whether A causes T or T causes A—or whether, indeed, there is even any causal relationship between the two [2]. This problem is expressed by the famous adage *correlation does not imply causation*. But what, then, does imply causation?

In order to infer the causal relationship between A and T, one will need either 1) prior knowledge of physics, which we assume for now is unavailable or 2) the ability to perform interventions on the variables in question. Following Peters, Janzing, et al. (2017), we could imagine a hypothetical scenario in which we perform an intervention on A by raising the grounds on which a city is built. Presumably, by raising the grounds high enough, we would observe a decrease in temperature. Conversely, constructing an enormous heating element to change the average temperature of the city would have no influence on its altitude. The former experiment would allow us to infer that A has a causal influence on T, while the latter experiment would only suffice to show the absence of a causal influence of T on A.

Examples like these can seem trivial, even facetious, at first sight. However, they touch upon deep limitations on what can and cannot be inferred from passive observation. As we will see in the following sections, there is a meaningful distinction between modeling *the data* and modeling *the world*. This distinction is perhaps especially important when dealing with autonomous systems embedded in the real world. Such systems should be able to act in an "imagined" space in order to not only predict the effects of their actions but to communicate the *reasons* behind those actions. Ideally, they should also have the capacity to introspect and reason about how the world *could have been* had they acted differently, such that they can effectively improve their performance in the future. Although it remains an open question how such systems might be realized, it is uncontroversial that causal modeling must play a central role in this regard.

[2] The correlation observed between A and T might also be due to an unobserved confounding variable causing both A and T.

3 Background

3.1 Causality

So far, we have discussed causal models only abstractly. In this section, we will introduce in detail a specific causal model known as a structural causal model, and we will see how it naturally lends itself to a graphical representation known as a causal graph. We will discuss the properties and assumptions that enable us to draw causal conclusions from data, and we will see how associational, interventional, and counterfactual reasoning delineate what has become known as the Pearl Causal Hierarchy. Finally, we will formally introduce the problem of causal discovery, thus laying the groundwork necessary for a discussion of our proposed method.

3.1.1 Structural Causal Models

Fundamental to the study of causality is a mathematical object known as a **structural causal model (SCM)** (Peters, Janzing, et al. 2017). An SCM consists of a set of variables $\mathcal{X} = \{X_1, ..., X_N\}$ and a set of functional relationships $\mathcal{F} = \{f_1, ..., f_N\}$ that determine how the value of one X_i depends on the value of other X_j. More formally, an SCM $\mathfrak{C}$ over a finite number N of random variables X_i is a set of structural assignments

$$X_i := f_i(\mathbf{PA}_{X_i}, U_i), \quad \forall i \in \{1, ..., N\}, \tag{3}$$

where $\mathbf{PA}_{X_i} \subseteq \mathcal{X} - X_i$ is a subset of variables or **parents** that cause the value of X_i. U_i is a random noise variable (also sometimes referred to as an **exogenous variable**) that acts as a stand-in for all causal factors directly relevant to X_i but which are not explicitly represented by $\mathfrak{C}$. The set of all noise variables is represented by $\mathcal{U} = \{U_1, ..., U_N\}$, and the variables are often assumed to be jointly independent (an assumption we will touch in more detail in §3.1.7). Where appropriate, we will denote an SCM by a 3-tuple $\mathfrak{C} := (\mathcal{X}, \mathcal{F}, P_{\mathcal{U}})$, thus gathering the variables, the functional relationships, and the joint distribution over the noises $P_{\mathcal{U}}$.

Note that the assignment in (3) should not be read as a mathematical equation but rather as a computational assignment. This avoids algebraic symmetry, which runs contrary to the notion of causality. For instance, assuming the relationship $X_2 := \beta X_1 + U_2$, observing $X_2 = 0$ implies $\beta X_1 = -U_2$ whereas *actively setting* X_2 to 0 tells us nothing about X_1 and U_2 (Pearl 2009). This relates closely to the concept of interventions, which we will discuss in more detail in §3.1.2.

An important property of SCMs is that they entail a probability distribution over the variables $\mathcal{X}$. This follows

from the fact that each $X_i \in \mathcal{X}$ is functionally dependent on its random noise variable U_i. We can thus talk about the joint distribution $P^{\mathfrak{C}}_{\mathcal{X}}$ over the variables in an SCM $\mathfrak{C}$, as well as any marginal $P^{\mathfrak{C}}_{X_i}$ or conditional $P^{\mathfrak{C}}_{X_i|X_j}$ we might be interested in. For instance, given a simple bivariate SCM $\mathfrak{C}$ with structural assignments

$$\begin{aligned} X_1 &:= U_1 \\ X_2 &:= 2X_1 + U_2 \end{aligned} \tag{4}$$

and $U_1, U_2 \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$, we have the marginals

$$\begin{aligned} P^{\mathfrak{C}}_{X_1} &= \mathcal{N}(0,1) \\ P^{\mathfrak{C}}_{X_2} &= \mathcal{N}(0,5). \end{aligned}$$

In this case, X_2 is functionally dependent on X_1 which also means that X_1 and X_2 are statistically dependent, which we denote by $X_1 \not\perp\!\!\!\perp X_2$. As we will see, the notion of dependence—or, more specifically, conditional *in*dependence—helps bridge the gap between model and data and is thus an indispensable tool in many—if not most—areas of causality.

Given that an SCM specifies a joint distribution over its variables $\mathcal{X}$, it is a generative model. To draw a sample from an SCM, we first realize a set of noise values from $P_{\mathcal{U}}$ after which we perform ancestral sampling of the variables in $\mathcal{X}$. However, a properly specified SCM is more powerful than a typical generative model. This is because we can think of it as simulating a small part of the world, including how the world would act in response to interventions.

3.1.2 Interventions

The power of causal models becomes apparent when we introduce the notion of **interventions**. In an intervention, we actively change the value of a variable X_i, thus inducing an alternate distribution over the variables in the associated SCM. This **interventional distribution** is due to one of two types of intervention:

1. **Hard intervention**: This changes the value of a variable X_i to a constant (i.e. $X_i := 2$), thus cancelling any influence from other variables in the SCM.

2. **Soft intervention**: This changes the mechanism by which the value of X_i is determined by its parents, thus leaving dependences intact but modifying the functional relationships.

Interventions are denoted by the *do*-operator. For instance, let $\mathfrak{C}$ be the SCM in (4). Then $do(X_2 := 2)$ denotes a hard intervention on X_2 while $P^{\mathfrak{C};do(X_2:=2)}_{\mathcal{X}}$ denotes the associated interventional joint distribution over the variables $\mathcal{X} = \{X_1, X_2\}$, where the superscript highlights that $\mathfrak{C}$ was the starting point (Peters, Janzing, et al. 2017). Looking at (4), we can conclude that

$$P^{\mathfrak{C};do(X_2:=2)}_{X_1} = \mathcal{N}(0,1),$$

while $P^{\mathfrak{C};do(X_2:=2)}_{X_2}$ collapses to a point mass on 2. If we now imagine randomizing the value of X_2 in a sequence of hard interventions, we will find that X_1 and X_2 go from being dependent in the **observational distribution** to being independent in the interventional distribution. This follows from the fact that X_2 does not determine X_1 in $\mathfrak{C}$, and thus any intervention we might perform on X_2 will leave X_1 unchanged. This highlights a crucial difference between statistical and causal models: Statistical models are silent on the question of causation and are thus fundamentally unable to predict the effect of interventions, while this is not the case for causal models. A causal model thus contains genuinely more information than a statistical one (Schölkopf et al. 2021).

An important principle to the notion of interventions is that of **independent causal mechanisms** (Schölkopf et al. 2021). Formally, it states that changing (by means of intervention) a mechanism $P(X_i \mid \mathbf{PA}_{X_i})$ does not change any other mechanisms $P(X_j \mid \mathbf{PA}_{X_j})$. Returning to the example of modeling altitudes and temperatures, the principle reduces to the statement that $P(A)$ and $P(T \mid A)$ are *informationally* independent. In other words, changing $P(A)$ has no impact on $P(T \mid A)$, since the latter models the physical mechanism generating temperature from altitude. Applied to larger models with more variables, the principle states that interventions are localized and thus do not affect the mechanisms by which the values of other variables are generated.

3.1.3 Counterfactuals

One of the more interesting aspects of causal models is their ability to represent counterfactuals. Informally, a counterfactual is an account of *what would have happened had something been different*. Borrowing another example from Peters, Janzing, et al. (2017), consider the following scenario: You are playing poker and are dealt the hand ♣J and ♣3. After evaluating your chances of winning the pot, you choose to fold. The flop is then presented by the dealer, revealing the cards ♣4, ♣Q, and ♣2. Upon seeing this, you regret folding your hand (five cards of the same suit is a flush, the fifth-highest hand in the game). Your regret in this scenario is a clear manifestation of counterfactual reasoning, in that had you not folded, your chances of winning would have been good. (On a side note, it can be argued that regret is always a manifes-

tation of counterfactual reasoning (Coricelli et al. 2010).)

In the context of SCMs, counterfactuals are represented by first taking into account what actually happened (i.e. you were dealt a ♣J and ♣3, and the flop was ♣4, ♣Q, and ♣2) by conditioning the SCM's noise variables on the set of values $\boldsymbol{x}$ observed for a subset of variables $\mathbf{X} \subseteq \mathcal{X}$. More formally, starting with SCM $\mathfrak{C} := (\mathcal{X}, \mathcal{F}, P_{\mathcal{U}})$, we first construct an alternate SCM $\mathfrak{C}_{\mathbf{X}=\boldsymbol{x}}$ such that

$$\mathfrak{C}_{\mathbf{X}=\boldsymbol{x}} := (\mathcal{X}, \mathcal{F}, P_{\mathcal{U}|\mathbf{X}=\boldsymbol{x}}).$$

Finally, a counterfactual is then computed by performing an intervention on a variable X_i (i.e. $do(X_i := 0)$ or "do not fold") in $\mathfrak{C}_{\mathbf{X}=\boldsymbol{x}}$, thus obtaining the counterfactual joint distribution $P_{\mathcal{X}}^{\mathfrak{C}_{\mathbf{X}=\boldsymbol{x}};do(X_i:=0)}$. Sticking to the poker example, this distribution summarizes the alternate reality in which the same hand and flop were dealt ($\mathbf{X} = \boldsymbol{x}$) but in which you did *not* fold ($do(X_i := 0)$).

Considering the importance of counterfactual reasoning in human cognition, it is interesting to consider how this capability might be implemented in larger machine learning systems. For now, however, this remains an open question and not one that we will explore further in this paper. For a more in-depth treatment on the topic of counterfactuals, including helpful examples, we refer the reader to Peters, Janzing, et al. (2017).

3.1.4 Pearl Causal Hierarchy

Up until now, we have discussed the capabilities afforded to us by causal models, including the ability to predict the effect of interventions and to reason counterfactually. We have also argued that causal models contain more information than statistical or purely predictive models. Interestingly, these capabilities sharply delineate a hierarchy known as the **Pearl causal hierarchy (PCH)**, also sometimes referred to as the **ladder of causation** (Bareinboim et al. 2021; Pearl and Mackenzie 2018). Informally, the hierarchy highlights the distinct nature of *seeing*, *doing*, and *imagining* by arguing how each activity demands increasingly detailed knowledge of the underlying data-generating process.

Prediction resides at the bottom of the PCH and is the purview of statistics and most current machine learning systems. A predictive model might estimate a conditional distribution $P(Y \mid X)$, thus mapping inputs X to target values Y. However, this is done by modeling correlations, and predictive models are thus unable to reason about the effect of interventions and, by extension, counterfactuals. For this reason, such models are said to be **associational** (Pearl 2009).

Interventional reasoning inhabits the second level of the PCH and requires a causal model. In general, it is impossible to infer a causal model on the basis of observational data alone [3]. It is thus a fundamentally different activity than associational reasoning and requires information not contained in the data. Interestingly, models capable of answering interventional questions are also capable of answering associational questions. This is a general property of the PCH: Each layer in the hierarchy subsumes the levels below it but requires increasingly fine-grained knowledge of the underlying data-generating process.

The final and highest level of the PCH is counterfactual reasoning. As mentioned in §3.1.3, reasoning counterfactually not only requires knowledge of the underlying causal relationships but also of the distribution over the noise variables. In other words, it requires a representation of a fully specified SCM, thus enabling the conditioning of the noise based on observed realizations of a subset of variables *and* a causal model to predict the effect of interventions. It should go without saying that such a model can not only answer counterfactual questions but also the associational and interventional questions of the lower levels.

The PCH is at the heart of many critiques aimed at the current limitations of machine learning. It offers a formal argument for why any system trained purely on observational data will be incapable of causal and counterfactual reasoning, regardless of the size and sophistication of the model architecture.

3.1.5 Causal Graphs

In §3.1.1, we introduced the mathematical definition of an SCM as a set of structural assignments over a set of variables. Importantly, this definition naturally lends itself to a graphical representation known as a **causal graph**. A causal graph $\mathcal{G} = (V, E)$ of an SCM $\mathfrak{C} := \{\mathcal{X}, \mathcal{F}, P_{\mathcal{U}}\}$ is a directed graph consisting of a set of nodes V, with each variable $X_i \in \mathcal{X}$ represented as a node in the graph. A directed edge $(i, j) \in E$ implies that $X_i \in \mathbf{PA}_{X_j}$, or in other words, if there is an edge from X_i to X_j in $\mathcal{G}$, then X_i causes X_j in $\mathfrak{C}$. Further, causal graphs are generally assumed to be acyclic, implying the restriction that variables can neither directly nor indirectly cause themselves. Although cyclic causal graphs are well-defined in some cases (Peters, Janzing, et al. 2017), in the following we restrict our attention to causal graphs belonging to the more common class of **directed acyclic graphs (DAGs)**.

[3] For an interesting exception to this, see Shimizu et al. (2006).

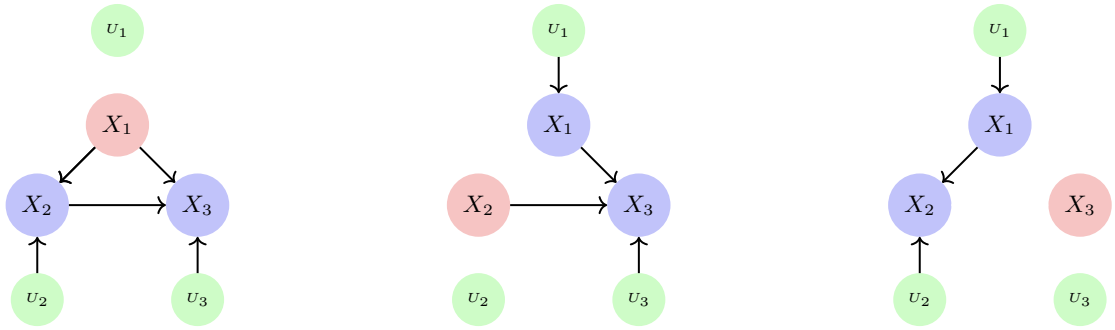


Figure 1: Graphical representation of hard interventions in a causal graph with three variables. When intervening on a variable (highlighted in red), any causal influence from its parents is removed. This is represented by a lack of incoming edges to the intervened variable.

Causal graphs allow one to reason more intuitively about the dynamics of a causal model. For instance, using a causal graph, one can determine, at a glance, whether two variables are dependent by simply inspecting if there is an edge between them or if they share a common ancestor [4]. They also lend themselves to intuitive representations of hard interventions. Given that a hard intervention on a variable X_i cancels any effects on X_i from its parents $\mathbf{PA}_{X_i}$, the associated graph is **mutilated** by removing edges (j, i) for all j. This is illustrated in Figure 1. Finally, the graphical perspective opens the door to graph-based algorithms like *d*-separation, which identify the conditioning set by which two variables X_i and X_j become conditionally independent. This is an indispensable tool in many scientific disciplines, where the conditional independences implied by a hypothesized causal graph can be used as a means of subjecting the hypothesis to falsification (Popper 2005). In other words, even in the absence of specified functional relationships $\mathcal{F}$ or a joint noise distribution $P_{\mathcal{U}}$, a causal graph on its own has **testable implications** (M. Glymour et al. 2016).

3.1.6 *d*-separation

The concept of ***d*-separation** has its origins in Bayesian networks and is a purely graphical procedure for determining conditional independence between variables. According to the rules of *d*-separation, an undirected path (i.e. a path in which one ignores the orientation of the edges) between variables A and B is blocked by conditioning on the variable C if and only if the following conditions are met:

1. A and B do not collide on C ($A \to C \leftarrow B$) nor any ancestors of C.
2. C is an intermediary node in a chain ($A \to C \to B$) or a fork ($A \leftarrow C \to B$) on the path between A and B.

If all paths between A and B are blocked in this way, the two variables are said to be *d*-separated. If there is more than one path connecting A and B, one will need a conditioning set Z of multiple variables to perform *d*-separation. We denote *d*-separation between A and B given conditioning set Z in a graph $\mathcal{G}$ by

$$A \text{ d-sep } B \text{ in } \mathcal{G} \mid Z.$$

As we will see in the following section, *d*-separation is the primary tool with which we formulate a crucial set of assumptions needed to connect a causal graph to its associated probability distribution.

3.1.7 Markovianity, Faithfulness & Sufficiency

It should be clear that causal graphs play an important role in causality. However, certain assumptions are needed in order to bridge the gap between conditional independences found in a distribution and the ones implied by a causal graph. Similarly, certain assumptions about SCMs (e.g. that the noise variables are jointly independent) do not always hold, and violations can sometimes make problems such as causal discovery much harder. Below, we list three of the most important of these assumptions.

Markovianity & Faithfulness

Markovianity and faithfulness are two complementary assumptions about the relationship between a causal graph and its associated probability distribution.

Markovianity states that a variable is independent of its non-descendants given its parents, which is equivalent to

[4] Assuming causal sufficiency. For more, see §3.1.7.

the statement that d-separation in the graph implies conditional independence in the distribution, i.e.:

$$A \text{ d-sep } B \text{ in } \mathcal{G} \mid Z \implies A \perp\!\!\!\perp B \mid Z.$$

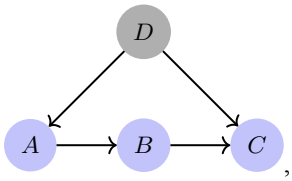
The assumption of **faithfulness**, on the other hand, reverses this implication by stating that if variables A and B are found to be conditionally independent given a set of variables Z in the distribution, A and B are d-separated by Z in the causal graph:

$$A \perp\!\!\!\perp B \mid Z \implies A \text{ d-sep } B \text{ in } \mathcal{G} \mid Z.$$

As we will see, faithfulness is an important assumption in the context of causal discovery. We will touch this point in more detail in §3.2.

Sufficiency

The assumption of **causal sufficiency** states that all noise variables in $\mathcal{U}$ are independent. In many settings, this is a strong assumption and is equivalent to the situation in which there are no **latent confounders**. A latent confounder is an unobserved variable that causes two or more observed variables. For instance, if A and B are observed and C is unobserved, C is a latent confounder in the causal graph $A \leftarrow C \rightarrow B$. When causal sufficiency is violated, conditional independences implied by a graph do not always translate to conditional independences in the distribution. To see this, consider the causal graph



,

where only the variable D is unobserved. In this case, the graph implies that $A \perp\!\!\!\perp C \mid B$. However, conditioning on B does not induce independence between A and C due to the latent confounder D. Therefore, this also constitutes a violation of Markovianity.

3.1.8 Identifiability

We conclude our introduction to the foundations of causal inference by introducing the concept of identifiability. The problem of identifiability arises naturally out of the following question: Given a joint distribution $P_{\mathcal{X}}$ induced by a causal model over variables $\mathcal{X}$, is the underlying causal graph **identifiable** from $P_{\mathcal{X}}$?

In §2, we touched on the fact that given a joint distribution $P_{A,T}$ over altitudes and temperatures, we cannot expect to be able to infer the causal direction ($A \rightarrow T$ or $A \leftarrow T$) in the absence of interventional data. However, if we find that $A \not\perp\!\!\!\perp T$, then, assuming causal sufficiency, we *can* infer the existence (though not the direction) of a causal relationship between A and T. This follows from **Reichenbach's common cause principle**: If two variables X and Y are statistically dependent, then there must be a third variable Z (which may or may not coincide with X or Y) that causally influences both (Peters, Janzing, et al. 2017). In the case of A and T, we can thus conclude from observational data that either $A \rightarrow T$ or $A \leftarrow T$. This ambiguity is sometimes represented by an undirected edge, in which case the associated graph is referred to as a **partially directed acyclic graph (PDAG)**.

When two or more causal graphs induce the same conditional independences, they are said to belong to a common **Markov equivalence class**. All graphs in a Markov equivalence class are characterized by having the same skeleton (i.e. the same set of edges if one ignores the direction) and the same immoralities (i.e. collider structures $X \rightarrow Z \leftarrow Y$, where X and Y do not share an edge) (Verma et al. 1991). A Markov equivalence class can thus be represented by a PDAG, or more precisely, a **completed partially directed acyclic graph (CPDAG)** in which all superfluous edges have been removed and all edges comprising immoralities have been oriented.

For a practical example, consider the graphs $A \rightarrow T$ and $A \leftarrow T$. The two graphs have the same skeleton and contain no immoralities. From this, we can conclude that they belong to the same Markov equivalence class and that they are therefore indistinguishable using observational data. On the other hand, the graph $X \rightarrow Z \leftarrow Y$ is the only graph in its Markov equivalence class, since no other graphs share the same skeleton and immoralities. This also means that $X \rightarrow Z \leftarrow Y$ is identifiable using only observational data (namely by conditional independence testing). Specifically, this is done by observing that X and Y are unconditionally independent ($X \perp\!\!\!\perp Y$) but become dependent when conditioning on Z ($X \not\perp\!\!\!\perp Y \mid Z$). Immoralities are the only graph structures with this property, and their edges can thus be oriented using observational data alone (M. Glymour et al. 2016).

3.2 Causal Discovery

Up until now, our primary focus has been on problems that arise from settings in which a causal graph has been given a priori, usually by human experts. Causal discovery, on the other hand, refers to the inference of causal structure in graphs whose nodes are given but where the edges are either partially or completely

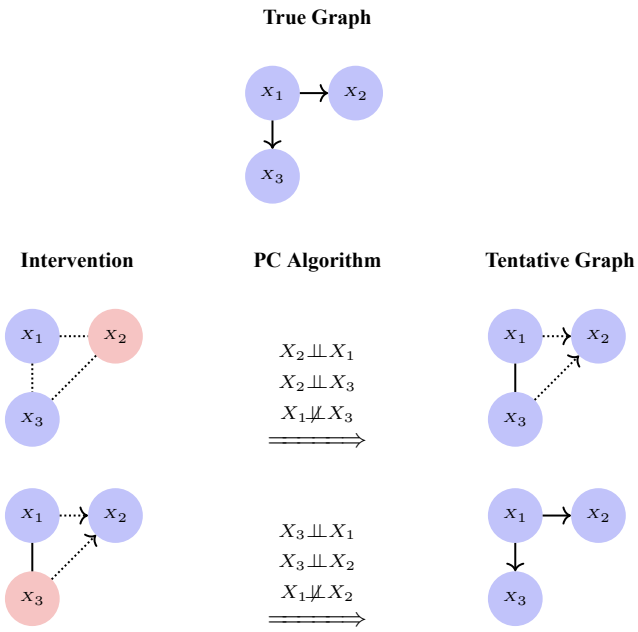


Figure 2: **Interventional causal discovery using the PC algorithm.** After each intervention, PC is used to determine independences in the resulting interventional distribution. For instance, after intervening on X_2, it is seen to be independent of X_1 and X_3 ($X_2 \perp\!\!\!\perp X_1$, $X_2 \perp\!\!\!\perp X_3$), meaning that 1) X_2 cannot be a parent of X_1 or X_3, and 2) either X_1 and / or X_3 are parents of X_2 in the graph or none of them are. Dashed lines represent tentative (not yet confirmed) causal relationships while solid lines represent confirmed relationships.

missing (C. Glymour et al. 2019). This problem is of particular interest in the field of machine learning, where the long-term ambition is to enable autonomous systems to construct deeper representations of the world than what is currently possible. In order for such systems to reach their full potential, however, they will have to learn most of these representations without human supervision and without being explicitly programmed.

To appreciate the difficulty of causal discovery, consider the number of possible DAGs for a graph of N nodes, denoted by $G(N)$. This is given by the following recurrence relation (Harary et al. 2014):

$$G(N) = \sum_{k=1}^{n} (-1)^{k+1} \binom{N}{k} 2^{k(N-k)} G(N-k)$$

This function is super-exponential in N and means that for a graph of 10 nodes, the number of possible DAGs is approximately 4.2×10^{18}. The problem of causal discovery, then, is to narrow down the often immense number of possible causal graphs to a single graph or a comparatively small subset of graphs.

The precision with which one should expect a causal discovery technique to predict the causal graph depends on the data available. In the observational setting, the causal graph is generally only identifiable up to a Markov equivalence class or, more precisely, its CPDAG. On the other hand, if enough interventional data is available, the true causal graph is always, in principle, identifiable.

Roughly speaking, causal discovery techniques can be categorized as **constraint-based** or **score-based**. Constraint-based techniques rely on conditional independence testing to infer the presence of edges in the underlying causal graph. While multiple constraint-based methods exist (Spirtes et al. 2000; Hyttinen et al. 2014; Richardson 2013), the PC algorithm (Spirtes et al. 2000) is perhaps the most well-known. At a high level, the PC algorithm works by initially assuming a fully connected graph. It then gradually prunes the edges between all pairs of variables (X, Y) based on independences induced by conditioning sets Z of increasing size. Assuming faithfulness between the distribution and underlying graph $\mathcal{G}$, this works since any such conditional independence implies that X d-sep Y in $\mathcal{G} \mid Z$, and therefore no direct edge between X and Y can exist. In its original formulation, the PC algorithm only works on observational data but can be easily extended to account for interventions as well. In this case, the algorithm is simply run on each interventional distribution while retaining information about previously discovered conditional independences. This is visualized for a graph of $N = 3$ nodes in Figure 2.

Another approach to the problem of causal discovery is that of score-based methods. These attempt to identify the true causal graph $\mathcal{G}^*$ by defining a score function $\mathcal{S}$, designed such that

$$\mathcal{G}^* \in \arg\max_{\mathcal{G}} \mathcal{S}(\mathcal{G}),$$

where $\mathcal{G}$ is an arbitrary **candidate graph** of the same cardinality as $\mathcal{G}^*$. Typically, the score function is formulated as a regularized likelihood function

$$\mathcal{S}(\mathcal{G}) = \max_{\theta} \mathbb{E}_{\mathcal{X} \sim P_{\mathcal{X}}} f_\theta(\mathcal{X}) + \lambda |\mathcal{G}|, \tag{5}$$

where $|\mathcal{G}|$ is the number of edges in the candidate graph, λ is a positive regularization coefficient, and f_θ is a density estimator parametrized by θ. Classical examples include the Greedy Equivalence Search (GES) and Greedy Interventional Equivalence Search (GIES) algorithms (Chickering 2002; Hauser et al. 2012). These work by initially assuming an empty graph, after which edges are iteratively added to greedily improve a score function based on the Bayesian Information Criterion (BIC). This is done using discrete search operators on predicted edges like *add*, *delete*, and *reverse*. In a second phase of the algorithm, edges are iteratively removed to improve the score.

A more recent example is the work of Zheng et al. 2018 who first formulated the causal discovery process as a continuous constrained-optimization procedure with immediate applications for neural causal discovery. Concretely, the authors showed how Lagrange-optimizing the likelihood in (5) subject to a continuous "DAGness" constraint suffices to uncover causal structure under the assumption of linear functional relationships. This work, in particular, has seen wide adoption in the nascent field of neural causal discovery (Ke et al. 2020; Brouillard et al. 2020; Kalainathan et al. 2018) and also serves as an important part of our proposed method, which we will introduce in §5. We will touch the topic of scoring the DAGness of a graph in more detail in §5.2.

3.3 Generative Adversarial Networks

A central field of study in machine learning and statistics is the problem of modeling a joint probability distribution given samples drawn from the distribution. Typically, this is done by minimizing some notion of divergence between the true distribution P_X and an approximate distribution Q_X, expressed in the form of a parametric model. For instance, the **variational autoencoder (VAE)** (Kingma et al. 2013) approximates P_X using neural networks by assuming a latent variable model $P_{X|Z}$ in which the observed variable X is generated on the basis of a lower-dimensional variable Z that allows for efficient approximation by simple parametric distributions.

An alternative approach due to Goodfellow et al. 2014 is to optimize a generative model using an **adversarial** training procedure. Concretely, the authors propose a two-pronged model architecture known as a **generative adversarial network (GAN)**, consisting of 1) a generator $\mathfrak{G}$ taking as input a random noise vector $\boldsymbol{z}$ from a prior distribution P_Z and 2) a discriminative network $\mathfrak{D}$ tasked with discriminating between samples from the true distribution P_{data} and samples from the approximate distribution P_g expressed by $\mathfrak{G}(\boldsymbol{z})$. During training, the discriminator and the generator are continually optimized for opposing goals: The discriminator for accurately distinguishing between samples from P_g and samples from P_{data}, and the generator for fooling the discriminator into classifying its "fake" samples as belonging to P_{data}. The authors show how this dynamic gives rise to the following minimax game:

$$\begin{aligned}\min_{\mathfrak{G}} \max_{\mathfrak{D}} \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log \mathfrak{D}(\boldsymbol{x})] + \\ \mathbb{E}_{\boldsymbol{z} \sim p_Z}[\log(1 - \mathfrak{D}(\mathfrak{G}(\boldsymbol{z})))]\end{aligned} \tag{6}$$

Here, $\mathfrak{D}(\boldsymbol{x})$ is the discriminator's predicted probability that the sample $\boldsymbol{x}$ belongs to P_{data} while $\mathfrak{G}(\boldsymbol{z})$ is the fake data point obtained by the generator when given $\boldsymbol{z} \sim P_Z$ as input. The authors then outline a simple algorithm for the training procedure. In short, for a dataset of m samples, training the discriminator involves maximizing the loss function in (7), while training the generator involves minimizing the loss function in (8).

$$\frac{1}{m} \sum_{i=1}^{m} \log \mathfrak{D}(\boldsymbol{x}^{(i)}) + \log(1 - \mathfrak{D}(\mathfrak{G}(\boldsymbol{z}^{(i)}))) \tag{7}$$

$$\frac{1}{m} \sum_{i=1}^{m} \log(1 - \mathfrak{D}(\mathfrak{G}(\boldsymbol{z}^{(i)}))) \tag{8}$$

Once the model has been trained, the discriminator is discarded, while the generator can be used to sample an endless number of synthetic data points that resemble those drawn from the target distribution P_{data}.

An intriguing property of generative adversarial networks is the fact that they sidestep the need for optimizing an explicit density function. This has given rise to the distinction between **explicit** and **implicit** generative models, of which GANs belong to the latter. This, however, also means that GANs often suffer from idiosyncratic training-related issues like mode collapse (Arjovsky and Bottou 2017) in which the generator learns to map all values of $\boldsymbol{z} \sim P_Z$ to the same output.

This has resulted in GANs gaining a reputation for being notoriously difficult to train, although more recent efforts like the Wasserstein training objective (Arjovsky, Chintala, et al. 2017) has managed to alleviate the problem somewhat.

GANs have shown remarkable results in modeling high-dimensional distributions, particularly those underlying large image datasets (Karras et al. 2019). Samples from GANs are typically characterized by sharpness and—in the case of natural images—photorealism to an extent that other generative models often do not. However, due to the implicit nature of the model, it is generally not possible to evaluate the likelihood of its generated samples and thus to gain insight into the learned distribution. Interestingly, recent research shows that samples from GANs even tend to be assigned low likelihood by models trained using explicit maximum likelihood methods (Grover et al. 2018), despite exhibiting superior qualitative characteristics.

4 Related Work

4.1 Neural Causal Models

The increasing interest in combining deep learning and causality has prompted the question of how to embed causal models within neural network-based architectures. Some efforts attempt to do this by explicitly modeling the conditional mass or density function of each variable given its causal parents using a dedicated neural network for each variable. In the case of discrete variables, this is easily accomplished by softmaxing the output of each neural network (Ke et al. 2020) into a PMF, while for continuous variables, more advanced methods like normalizing flows (Kobyzev et al. 2020) or VAEs (Kingma et al. 2013) are typically applied (Pawlowski et al. 2020; Brouillard et al. 2020). In this section, we specifically focus on the work of Xia et al. 2021, who formulated an alternative architecture known as a **neural causal model (NCM)** in which densities are modeled implicitly by taking as input a noise variable from a prior distribution in addition to the similarly computed outputs for the variable's ancestors.

More formally, Xia et al. 2021 define an NCM over variables $\mathcal{X} = \{X_1, ..., X_N\}$ as a set of neural networks $\mathcal{F} = \{f_{\theta_1}, ..., f_{\theta_N}\}$ parametrized by $\theta_1, ..., \theta_N$ and a prior joint noise distribution $P_{\mathcal{U}}$ over $U_1, ..., U_M$ with $M \leq N$ and each $U_i \perp\!\!\!\perp U_j$ for all pairs (i, j). Each f_{θ_i} receives exactly one U_j as part of its input, meaning that for $M < N$, one U_j will act as input to multiple f_{θ_i} so as to model the effect of latent confounders. On the other hand, causal sufficiency is assumed in the case where $M = N$, such that f_{θ_i} maps $U_i \cup \mathbf{PA}_{X_i}$ to a direct realization of the random variable X_i. According to the definition, each P_{U_i} should follow a uniform distribution in the range $[0, 1]$. We believe this should not be seen as a hard restriction on P_{U_i} but rather as one out of many possible distributions; the important point being that each P_{U_i} should follow a fixed and preferably simple parametric distribution. Further, the authors argue that such a formulation can be used to approximate any SCM as long as the f_{θ_i}'s are sufficiently expressive. This follows from the well-known fact that neural networks act as universal function approximators when given enough capacity in their hidden layers (Hornik 1991).

Modeling interventional distributions in an NCM is simply a matter of setting the target of the intervention X_i to a desired value x and performing ancestral sampling of the remaining variables. In fact, NCMs only differ from SCMs in the fact that they are parametrized by neural networks with identical distributions over each noise variable. In §5, we explore a method by which training an NCM can serve as a method of causal discovery.

4.2 DSDI

One example of a neural causal discovery method is the DSDI (**D**ependency **S**tructure **D**iscovery from **I**nterventions) framework proposed by Ke et al. (2020). This method relies on what the authors call **structural** and **functional** parameters to predict the causal structure underlying a set of categorical variables $\mathcal{X} = \{X_1, ..., X_N\}$. The structural parameters $\gamma \in \mathbb{R}^{N \times N}$ form the backbone of the model's **edge beliefs**, encoded by $\sigma(\gamma)$, where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function, such that $\sigma(\gamma_{ij})$ gives the model's predicted probability of the presence of the edge (i, j). The functional parameters $\theta_1, ..., \theta_N$ parametrize the MLPs $f_{\theta_i} : \mathbb{R}^N \to \mathbb{R}^K$ modeling the conditional distribution of each variable given its predicted parents, where K denotes the number of categories for the variable in question.

Although this formulation looks similar to that of an NCM, it is slightly different. In an NCM, the output of one f_{θ_i} serves as input to all of its children f_{θ_j}. In DSDI, the inputs to each f_{θ_i} are the actual observed values of its parent variables found in the dataset. This gives rise to a distinction between what we call a **self-contained NCM** and a **non-self-contained NCM**: A self-contained NCM follows the definition outlined in §4.1 and is characterized by having the output of each f_{θ_i} act as the input to its immediate descendants. In contrast, each f_{θ_i} in a non-self-contained NCM receives as input observed samples from the true underlying SCM. This makes it

possible for non-self-contained NCMs to contain cycles in their associated graph structure. The distinction between self-contained and non-self-contained NCMs is thus only a matter of the values input to each f_{θ_i} and not of architecture. (Note that while DSDI relies on the use of non-self-contained NCMs during training, the resulting model can act as a self-contained NCM insofar as the final predicted edge structure is acyclic. The fact that each f_{θ_i} does not take a noise variable as input but instead outputs a categorical distribution is of less importance.)

The training procedure in DSDI proceeds in phases. Initially, randomly drawn observational samples from the true underlying causal model are used to update $\theta_1, ..., \theta_N$ using maximum likelihood estimation. Multiple graphs $\{\mathcal{G}^{(1)}, ..., \mathcal{G}^{(M)}\}$ are then sampled according to the model's current edge beliefs $\sigma(\gamma)$, after which an intervention (soft or hard) is performed on a single variable, picked at random. The likelihoods of the observed variables are computed on the basis of a softmax over each of the M outputs from $f_{\theta_1}, ..., f_{\theta_N}$, and the structural parameters γ are then updated accordingly. This is done using a REINFORCE-like gradient estimator (Williams 1992; Bengio et al. 2019), since each graph is obtained through (non-differentiable) Bernoulli sampling over each edge belief. This process then repeats. Notably, DSDI also incorporates a heuristic for inferring causal structure when the target of an intervention is unknown. The method, therefore, supports a wide range of intervention types (soft, hard, known, and unknown).

When benchmarked on multiple datasets from the BnLearn repository [5], the method performs favourably compared to previous methods, although it begins to encounter difficulties for graphs with $N > 10$ nodes. In the case of partial graph recovery, where a subset of the edges are given at the outset, the method scales to ~ 50 variables.

4.3 Two-Phase DAG Sampling

One limitation of the DSDI framework outlined in the previous section is the possibility of sampling cyclic graphs from the model's edge beliefs. Although the authors attempt to circumvent this by regularizing the loss function with a term penalizing two-hop cycles, it does not fully alleviate the issue. To address this problem, Scherrer et al. (2021) recently introduced a novel graph sampling method which similarly relies on a matrix of edge beliefs but guarantees acyclicity. The method relies on the fact that the adjacency matrix of an acyclic graph can be sorted in topological order (i.e. such that for all edges (i, j), node i precedes node j in the ordering). For an adjacency matrix A with A_{ij} indicating the presence of an edge from node i to node j, such an ordering corresponds to A being upper triangular with zeros along the diagonal.

The DAG sampling procedure takes advantage of this property by proceeding in two phases: First, a topological order is sampled according to the model's edge beliefs $\sigma(\gamma)$. This is done by identifying the inverse of the maximum probability of each node having an incoming edge. For node i, this probability p_i is defined as

$$p_i = \min_j 1 - \sigma(\gamma_{ji}).$$

Intuitively, the smaller the p_i, the less likely it is that X_i is a root node, since high probability is placed on at least one incoming edge to X_i. Sampling is then performed over the softmaxed probabilities $p'_1, ..., p'_N$ with an optional temperature parameter T controlling the entropy of the resulting distribution, i.e.

$$p'_i = \frac{\exp(p_i/T)}{\sum_{j=1}^{N} \exp(p_j/T)}.$$

The first sample from this distribution gives the root node of the sampled graph. When a node i has been sampled, the ith row and ith column are removed from γ, and the procedure then continues until the node set is exhausted. Once a topological order has been sampled, the second phase permutes the matrix of edge beliefs accordingly and forcibly sets the resulting matrix γ' to be upper triangular. Finally, a DAG can be obtained by sampling edges (i, j) using Bernoulli sampling of the remaining edge beliefs $\sigma(\gamma'_{ij})$.

5 Adversarially Trained Neural Causal Models

We now present a novel method of interventional, score-based causal discovery with continuous variables, borrowing elements from the recent work of Scherrer et al. (2021) and Ke et al. (2020). The idea behind the method was inspired by the observation that an ability to sample DAGs from a matrix of edge beliefs enables one to construct a self-contained NCM, even when the edge beliefs are characterized by high entropy and potential cycles. This enables a new mode of training in which the output of a neural network $f_{\theta_i} : \mathbb{R}^{N+1} \to \mathbb{R}$ modeling the conditional distribution of variable X_i given its predicted parents and a noise variable, can reliably serve as input to the functions f_{θ_j} modeling the predicted children of variable X_i. In this formulation, the only external input to the NCM are samples from a prior noise

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

distribution $P_{\mathcal{Z}}$.

In order to train this model, we follow the generative adversarial framework proposed by Goodfellow et al. (2014). This should seem natural given the similarities between a GAN generator and a self-contained NCM. (Each is a generative model receiving as input only a random noise vector.) During training, the model's discriminator attempts to distinguish samples from the NCM and samples from a ground truth SCM. This is done using both observational and interventional data.

We make the following assumptions:

- Randomized interventional data is available for all variables $X_i \in \mathcal{X}$
- Causal sufficiency (i.e. no latent confounders)
- All $X_i \in \mathcal{X}$ are continuous
- All $U_i \in \mathcal{U}$ are Gaussian

Given that the NCM obeys the graph structure sampled from the matrix of edge beliefs, we operate under the assumption that the edge beliefs that maximize the NCM's ability to "fool" the discriminator across all interventional distributions are the ones corresponding to the actual edges. (A proof of this would be valuable but is left as future work.) Although causal discovery using adversarial training has been attempted by others (Kalainathan et al. 2018), to the best of our knowledge this has only been done in the context of observational data and non-self-contained NCMs. The question, then, is two-fold: 1) Whether or not adversarial training can serve as a method of optimizing self-contained NCMs, and 2) whether a matrix of accurate edge beliefs arises as a byproduct of successfully fooling the discriminator.

We find that adversarial training can consistently identify causal structure in linear settings with $N \leq 8$ but that it struggles to correctly identify graphs with high edge densities as well as in settings with nonlinear data. We explore the importance of regularization in an ablation study, and we evaluate our method on synthetic datasets of linear and nonlinear data with additive noise. Finally, we offer some reflections on the benefits and drawbacks of our method as well as on potential areas of future work.

5.1 Architecture

We parametrize an NCM $\mathfrak{N}$ using a DAG $\mathcal{G}$ and a set of neural networks $\mathcal{F} = \{f_{\theta_1}, ..., f_{\theta_N}\}$. Each $f_{\theta_i} : \mathbb{R}^{N+1} \to \mathbb{R}$ is an MLP parametrized by θ_i consisting of one 32-unit hidden layer with Leaky ReLU activation and no activation applied at the output layer.

The j'th entry of the input vector $\boldsymbol{v}_i \in \mathbb{R}^{N+1}$ to f_{θ_i} is 0 if $X_j \notin \mathbf{PA}^{\mathcal{G}}_{X_i}$, i.e. if X_j is not a parent of X_i in $\mathcal{G}$, else it contains the output of $f_{\theta_j}(\boldsymbol{v}_j)$. The last entry in $\boldsymbol{v}_i$ contains a realization of a random noise variable Z_i sampled from a prior noise distribution P_{Z_i}. Thus, letting A denote the adjacency matrix of graph $\mathcal{G}$, with A_{ij} indicating the presence of the edge (i, j), we have

$$\boldsymbol{v}_i = \begin{bmatrix} A_{1i} f_{\theta_1}(\boldsymbol{v}_1) & \cdots & A_{Ni} f_{\theta_N}(\boldsymbol{v}_N) & Z_i \end{bmatrix}.$$

Importantly, P_{Z_i} does not need to follow the true noise distribution P_{U_i} of the SCM being modeled, granted that f_{θ_i} is sufficiently expressive. In other words, if f_{θ_i} models the causal mechanism generating the value of a root node X_i with true distribution P_{X_i}, f_{θ_i} should learn a mapping from arbitrary noise values (i.e. ones following another distribution) to P_{X_i}. In this work, we limit our attention to the setting in which the prior joint noise distribution $P_{\mathcal{Z}}$ is uniform in the range $[0, 1]$ (this follows the definition of an NCM by Xia et al. 2021), whereas $P_{\mathcal{U}}$ is a non-isotropic Gaussian.

To represent our edge predictions, we follow the approach in DSDI (§4.2) and rely on a soft adjacency matrix $\gamma \in \mathbb{R}^{N \times N}$ such that

$$\sigma(\gamma_{ij}) = \frac{1}{1 + \exp(-\gamma_{ij})}$$

gives the predicted probability of an edge from X_i to X_j. Ideally, this probability will reflect the true underlying causal relationship once the model has been trained. In order to guarantee that the functional relationships of each NCM are well-defined, it is important to ensure that every graph sampled from $\sigma(\gamma)$ is acyclic. To do so, we follow the two-phase DAG sampling procedure outlined in §4.3 and denote this by

$$\mathcal{G} \overset{\text{DAG}}{\sim} \sigma(\gamma).$$

In a small departure from the work of Scherrer et al. 2021, we take a slightly different approach to sampling the edges of the upper triangular matrix γ' arrived at by the end of the first phase. Where Scherrer et al. 2021 choose to include an edge based on Bernoulli sampling, we instead make use of the differentiable Straight-Through Gumbel-Softmax trick (Jang et al. 2016) applied on each $\sigma(\gamma'_{ij})$ and $1 - \sigma(\gamma'_{ij})$. This obviates the need for REINFORCE-like gradient estimators to propagate gradients back to the edge beliefs.

By guaranteeing the DAGness of graphs sampled from $\sigma(\gamma)$, the NCMs will always be self-contained and thus give easily computed outputs on the basis of only a

noise vector $\boldsymbol{z} \in \mathbb{R}^N$. We feel this property is worth emphasizing. Given that other methods in the literature typically rely on f_{θ_i} to predict the value of X_i given variables $\mathbf{X} \subseteq \mathcal{X} - X_i$ from the true underlying SCM, such NCMs are well-defined even in the presence of cycles in the sampled graph structure. In contrast, guaranteeing DAGness of the sampled graphs ensures valid graph structures throughout training and enables an architecture based on edge beliefs in which the output of one f_{θ_i} can reliably serve as input to another f_{θ_j}. In the following, we refer to the output of an NCM $\mathfrak{N}$ given a vector of noise values $\boldsymbol{z} \in \mathbb{R}^N$ and parametrized by a DAG $\mathcal{G}$ as $\mathfrak{N}^{\mathcal{G}}(\boldsymbol{z})$.

Using the NCM framework, it is easy to model interventions. Concretely, if we wish to sample from the interventional distribution $P^{\mathfrak{N}^{\mathcal{G}};do(X_i=x)}$, we simply skip computing $f_{\theta_i}(\boldsymbol{v}_i)$ and instead set its associated value to x. Any child X_j of X_i in $\mathcal{G}$ will then have the i'th component of its input vector $\boldsymbol{v}_j$ set to x. This construction means that for an intervention on variable X_i, neither θ_i nor the edge beliefs on incoming edges to X_i are updated for such samples. This is a desirable—and likely necessary—property, since neither of those parameters were responsible for generating the data in question.

5.2 Loss & Regularization

We take inspiration from Goodfellow et al. (2014) and introduce a discriminator $\mathfrak{D} : \mathbb{R}^{2N+1} \to (0, 1)$, implemented as an MLP with two 128-unit hidden layers with Leaky ReLU activations and a sigmoid activation at the output layer. Similar to the traditional GAN setup, $\mathfrak{D}$ is trained to distinguish fake samples (i.e. samples from $\mathfrak{N}$) from real samples (i.e. samples from a ground truth SCM $\mathfrak{C}$) by maximizing

$$\frac{1}{m}\sum_{i=1}^{m} \log \mathfrak{D}(\boldsymbol{x}^{(i)} \,||\, \boldsymbol{h}^{(i)}) + \log(1 - \mathfrak{D}(\mathfrak{N}^{\mathcal{G}^{(i)}}(\boldsymbol{z}^{(i)}) \,||\, \boldsymbol{h}^{(i)})), \tag{9}$$

where $\boldsymbol{x}^{(i)} \in \mathbb{R}^N$ is a sample drawn from $\mathfrak{C}$, $\boldsymbol{h}^{(i)} \in [0, 1]^{N+1}$ is a one-hot vector in which the j'th entry is 1 if the sample is from either $P_{\mathcal{X}}^{\mathfrak{C};do(X_j=x)}$ or $P_{\mathcal{X}}^{\mathfrak{N};do(X_j=x)}$, and $||$ denotes concatenation. Samples from either observational distribution is represented by a 1 in entry $N + 1$ of $\boldsymbol{h}^{(i)}$. The parameters γ and $\theta_1, ..., \theta_N$ of $\mathfrak{N}$ are trained to optimize the opposing objective by minimizing

$$\frac{1}{m}\sum_{i=1}^{m} \log(1 - \mathfrak{D}(\mathfrak{N}^{\mathcal{G}^{(i)}}(\boldsymbol{z}^{(i)}) \,||\, \boldsymbol{h}^{(i)})). \tag{10}$$

We found appropriate regularization to be important for aligning the model's edge beliefs with the edges of the ground truth DAG $\mathcal{G}^*$. Concretely, we follow others in the literature and include two regularization terms:

1. An edge penalty J_E, encouraging sparsity in the predicted graphs
2. A cycle penalty J_{DAG} on $\sigma(\gamma)$, discouraging high edge beliefs on mutually exclusive edges

The edge penalty J_E is defined as the L1 norm of the mean of the adjacency matrices sampled from the matrix of edge beliefs in each iteration of training. (We introduce the training procedure in more detail in §5.3.) We weight J_E by a hyperparameter λ_E.

The cycle penalty uses the work of Zheng et al. (2018), who cast the problem of scoring the DAGness of a nonnegative weighted adjacency matrix $A \in \mathbb{R}_{\geq 0}^{N \times N}$ as a continuous optimization problem. Concretely, the authors show how the equation

$$\text{Tr}\, e^A - N = 0$$

is satisfied if and only if A is acyclic. This follows from the fact that the positivity of entry A_{ij}^k implies the existence of a length-k path between node i and node j. Since

$$e^A = \sum_{i=0}^{\infty} \frac{1}{i!} A^i = I + A + \frac{1}{2}A^2 + \cdots,$$

it follows that $\text{Tr}\, e^A = N$ for acyclic graphs only. Importantly, this expression is smooth and has easy-to-compute derivatives suitable for gradient descent. The requirement of nonnegativity of the adjacency matrix poses no problem in the case of our method, since $0 \leq \sigma(\gamma_{ij}) \leq 1$. We thus define the cycle penalty as

$$J_{\text{DAG}} = \text{Tr}\, e^{\sigma(\gamma)} - N$$

and weight it by the hyperparameter λ_{DAG}.

Interestingly, we found J_{DAG} to be detrimental to our model's predictions when applied throughout the training process, as it would occasionally push the edge beliefs on true edges towards 0 in the beginning stages of training. However, we did find it to be crucial to our model's predictions when applied later in the training process. Conversely, J_E seems to be mostly beneficial in the beginning of training but can prevent true edges from attaining a sufficiently high edge belief in the latter stages. To account for this, we linearly decay the value of λ_E as a function of the number of epochs, while we increase the value of λ_{DAG} in a similar way. Concretely, we set

$$\lambda_E = \max(0.01, 0.1 - \frac{l}{k} \cdot 0.1)$$
$$\lambda_{\text{DAG}} = 0.1 \cdot \frac{l}{k},$$

where l is the current epoch and k is the total number of epochs. λ_E is lower-bounded at 0.01 as we would occasionally observe erroneous edges attaining high edge beliefs in the very last stages of training (i.e. when the edge penalty approached 0). We provide a more detailed account of the effect of regularization on our model's performance in §5.4.3.

5.3 Training Procedure

True to the GAN framework, we train the model in an alternating fashion, switching between updating γ and $\theta_1, ..., \theta_N$ of the NCM and the parameters of $\mathfrak{D}$. We assume a dataset from $\mathfrak{C}$ consisting of 500 samples for the observational distribution as well as for each interventional distribution. We use a batch size of 256, which is in line with the approach taken by Ke et al. 2020.

The process of training $\mathfrak{N}$ and γ deviates slightly from how one would optimize the generator in a regular GAN. To see why, consider again the formulation of the minimax game in 6. From the perspective of $\mathfrak{N}$ and γ, the quantity to minimize can be expressed as the following nested expectation:

$$\mathbb{E}_{\boldsymbol{z}\sim P_Z}[\mathbb{E}_{\mathcal{G}\overset{\text{DAG}}{\sim}\sigma(\gamma)}[\log(1-\mathfrak{D}(\mathfrak{N}^{\mathcal{G}}(\boldsymbol{z}) \mid\mid \boldsymbol{h})) + \lambda_E J_E + \lambda_{\text{DAG}} J_{\text{DAG}}]].$$

This motivates us to alternate the training process between optimizing the outer expectation by training the θ_i's of the NCM and optimizing the inner expectation by updating the edge beliefs. In a slight abuse of terminology, we choose to refer to two full iterations over the dataset, one updating the θ_i's and the other updating γ, as one epoch.

The training procedure is structured as follows (regardless of the parameters being updated): First, we sample a batch of noise vectors $\{\boldsymbol{z}^{(1)}, ..., \boldsymbol{z}^{(m)}\} : \boldsymbol{z}^{(i)} \in \mathbb{R}^N$ from the prior joint noise distribution $P_{\mathcal{Z}}$. We then sample a batch of candidate graphs $\{\mathcal{G}^{(1)}, ..., \mathcal{G}^{(m)}\}$ with each $\mathcal{G}^{(i)} \overset{\text{DAG}}{\sim} \sigma(\gamma)$ sampled from the edge beliefs with temperature $T = 0.1$. The edge beliefs are initialized to

$$\sigma(\gamma_{ij}) = 0.5 \qquad \forall (i,j) : i \neq j$$
$$\sigma(\gamma_{ij}) = 0 \qquad \forall (i,j) : i = j.$$

For each sample in the batch, we then pick, at random, a variable on which to intervene (if any). For instance, intervening on X_i in the j'th sample means that we mutilate $\mathcal{G}^{(j)}$ such that any incoming edges to X_i are removed. The NCM then sets the value of X_i to a sample from $\mathcal{N}(2, 1)$ after which it performs ancestral sampling of the remaining variables in accordance with the mutilated graph. Doing this for all m samples, we obtain

$$\mathfrak{N}^{\mathcal{G}^{(1)}}(\boldsymbol{z}^{(1)}), ..., \mathfrak{N}^{\mathcal{G}^{(m)}}(\boldsymbol{z}^{(m)}).$$

Each sample $\mathfrak{N}^{\mathcal{G}^{(i)}}(\boldsymbol{z}^{(i)})$ is then input to $\mathfrak{D}$ along with the one-hot vector $\boldsymbol{h}^{(i)}$ indicating from which interventional distribution the sample was drawn. To update the parameters of the MLPs, we minimize the loss in (10), while updating the edge beliefs involves minimizing the same expression but with added regularization terms:

$$\frac{1}{m}\sum_{i=1}^{m} \log(1 - \mathfrak{D}(\mathfrak{N}^{\mathcal{G}^{(i)}}(\boldsymbol{z}^{(i)}) \mid\mid \boldsymbol{h}^{(i)})) + \lambda_E J_E + \lambda_{\text{DAG}} J_{\text{DAG}}.$$

As previously mentioned, J_E is computed by taking the L1 norm of the mean adjacency matrix sampled from the edge beliefs in the given batch.

In each iteration, we also update the parameters of $\mathfrak{D}$. This is done by first drawing a new batch of samples from the NCM in the way just described. Then, m samples are drawn at random from the dataset generated by the SCM, and the parameters are updated by maximizing the loss in (9).

We set the number of epochs to train the model k as a function of the number of nodes. In our experiments, we set the number of epochs required to predict the structure of a graph of N nodes as

$$k = \max(500, 100 \cdot N).$$

The lower bound of 500 is beneficial when N is small. We also employ early stopping if convergence is detected at any point during training. We define convergence as

$$\sigma(\gamma_{ij}) > 0.2 \vee \sigma(\gamma_{ij}) < 0.01 \quad \forall (i,j) : i \neq j$$

or, in words, as the situation in which all edge beliefs off the main diagonal are either above 0.2 or below 0.01. (The edge beliefs on the main diagonal are always 0 as they would otherwise violate acyclicity.) Finally, we set the learning rate for the parameters of $\mathfrak{D}$ to 10^{-4}, while we set it to 10^{-3} for the θ_i's. The learning rate for the edge beliefs is initially set to $5 \cdot 10^{-4}$ but increases linearly towards $5 \cdot 10^{-3}$ throughout training.

We illustrate the training procedure in Figure 3.

5.4 Experiments

We evaluate our proposed method across synthetic datasets drawn from linear as well as nonlinear SCMs.

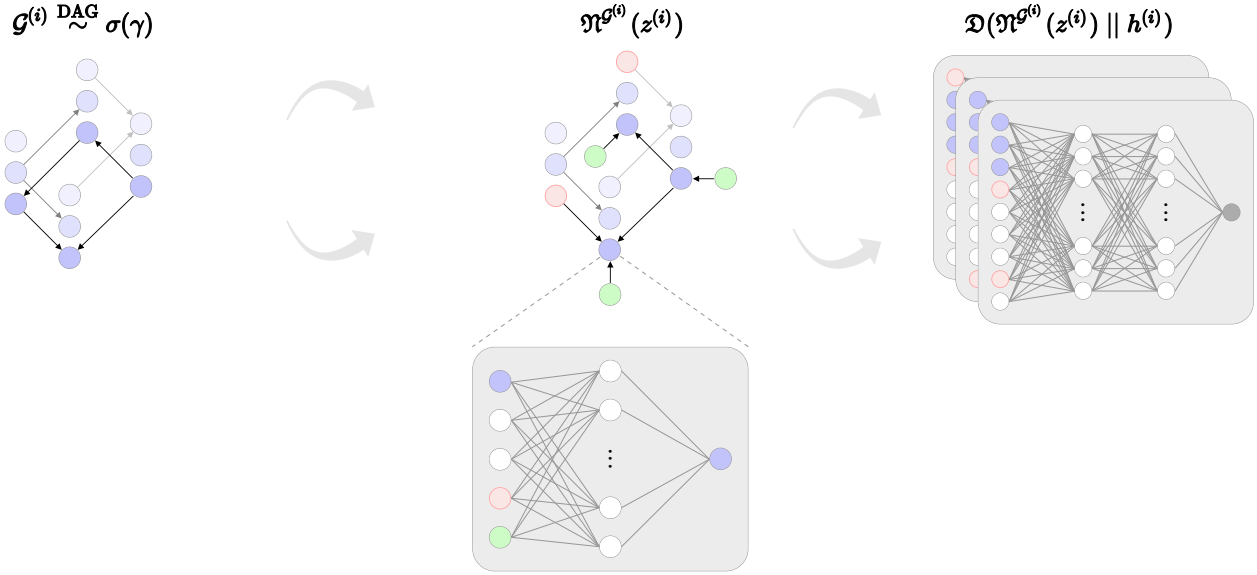


Figure 3: During each iteration of training, m DAGs $\{\mathcal{G}^{(1)}, ..., \mathcal{G}^{(m)}\}$ are sampled from the matrix of edge beliefs $\sigma(\gamma)$ (left). For each of the m samples in a batch, a noise vector $\boldsymbol{z}^{(i)} \in \mathbb{R}^N$ is drawn from a prior joint distribution $P_{\mathcal{Z}}$ with the j'th element $\boldsymbol{z}_j^{(i)}$ (green) being input to the MLP f_{θ_j} modeling node X_j (bottom). The value of intervened variables (red) are not computed using their associated MLP but are set to a random sample drawn from $\mathcal{N}(2, 1)$. Once a value has been obtained for every $X_j \in \mathcal{X}$, the values are input to the discriminator (right) concatenated with a one-hot vector indicating which distribution the sample came from.

All datasets were generated on the basis of **structured graphs** as well as **random graphs** of varying edge densities. Structured graphs are graphs following a predefined edge structure, e.g. a chain ($X_1 \to X_2 \to X_3$) or a collider ($X_1 \to X_2 \leftarrow X_3$). We follow the approach of Ke et al. (2020) and evaluate our method on six different structured graphs, shown in Appendix A.3. Random graphs were generated according to the **Erdős–Rényi (ER) model**, which samples graphs of fixed cardinality by assigning to each possible edge a probability of inclusion. To vary the edge density, we set $e = 1$ or $e = 2$ and sample graphs such that $N \cdot e$ gives the expected number of edges in the graph. An ER-1 graph thus refers to a randomly generated graph with N expected edges. (For more details, see Appendix A.2.)

We measure the error of our model's predictions using the **structural Hamming distance (SHD)** (Tsamardinos et al. 2006) between the true graph and the predicted graph. The SHD between two graphs $\mathcal{G}$ and $\mathcal{H}$ measures the number of edge flips, insertions, or deletions required to transform $\mathcal{G}$ into $\mathcal{H}$ (or vice-versa). It is thus a non-negative integer with 0 being a perfect score. We compute the SHD using the ground truth adjancency matrix A^* and the predicted adjacency matrix A. We define the predicted adjacency matrix as

$$A_{ij} = \begin{cases} 0 & \text{if } \sigma(\gamma_{ij}) \leq t \\ 1 & \text{otherwise} \end{cases},$$

where t is a threshold value. In our experiments, we set $t = 0.2$. Due to computational limitations, we limit our focus to graphs of $N \leq 10$ nodes. Our implementation is CPU-intensive due to the sequential nature of both the DAG sampling as well as the computation of the NCM outputs. Since the neural networks are small (at most two hidden layers of 128 units), using a GPU provider such as Google Colab or Kaggle dramatically slowed down training. All experiments were thus performed locally on a 2.6 GHz 6-Core Intel Core i7. Due to time constraints, we leave the development of a more efficient implementation as future work.

5.4.1 Synthetic Linear Data

Given a structured or random ground truth DAG $\mathcal{G}^*$, the synthetic linear datasets were generated following the approach of Brouillard et al. (2020). Concretely, we set the value of each root node to

$$\mathcal{N}(0, \sigma^2) : \sigma^2 \sim \mathcal{U}(1, 2), \tag{11}$$

where $\mathcal{U}$ is the uniform distribution, sampled for every root node in the graph. For all non-root nodes X_i, we set the value to

$$X_i := \boldsymbol{w}_i^{\mathrm{T}} X_{\mathbf{PA}_i^{\mathcal{G}^*}} + U_i, \tag{12}$$

where $\boldsymbol{w}_i$ is a column vector of coefficients and $X_{\mathbf{PA}_i^{\mathcal{G}^*}}$ is a column vector containing the values of the parents of

	Structured						Random	
N	Chain	Collider	Tree	Bidiag	Jungle	Full	ER-1	ER-2
2	0 ± 0	-	-	-	-	-	-	-
4	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	-
6	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0.2 ± 0.4	0.4 ± 0.8
8	0 ± 0	0.6 ± 0.8	0 ± 0	0.8 ± 0.9	0 ± 0	3.2 ± 2.2	0.6 ± 0.6	1.4 ± 2.8
10	0 ± 0	2.4 ± 3.8	0.8 ± 1.6	1.4 ± 2.8	0.8 ± 1.2	7.4 ± 3.9	0.4 ± 0.5	1.2 ± 1.9

Table 1: Mean SHD (lower is better) and standard deviation for our proposed method on linear SCMs with both structured and random graphs. Each benchmark was obtained by running our method 5 times on SCMs with randomly generated parameters.

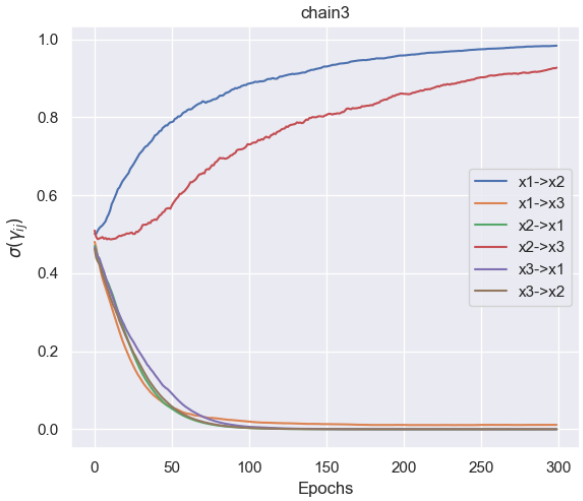


Figure 4: Progression of edge beliefs when training our model on a 3-node chain graph ($X_1 \to X_2 \to X_3$).

X_i in $\mathcal{G}^*$. The coefficients in $\boldsymbol{w}_i$ are mutually independent and sampled uniformly from $[-1, -0.25] \cup [0.25, 1]$, thus ensuring that no $\boldsymbol{w}$ is close to the zero-vector, which would (in the limit) violate faithfulness. U_i is obtained according to (11). Finally, we model hard interventions on a variable X_i by replacing its definition in (12) by the marginal $\mathcal{N}(2, 1)$. Interventions are thus randomized rather than set to the same constant in each sample.

Table 1 shows the SHD of our proposed method on structured and random graphs of varying cardinalities and edge densities. We find that our method performs well on certain graphs structures of up to $N = 10$ nodes, specifically chains, trees, jungles, and ER-1 graphs. While we see perfect predictions on a majority of graphs with $N \leq 6$ nodes, performance begins to deteriorate for denser graphs with $N = 8$. With $N = 10$, we find that the method begins to occasionally suffer from "catastrophic failure." For instance, of the 5 runs on the bidiagonal graph with 10 nodes, four had an SHD of 0 while one had an SHD of 7. We observe a similar phenomenon for the collider graph with 10 nodes. It might thus be advisable to use the algorithm in an ensemble-like fashion by running it multiple times and picking the most commonly predicted graph. We leave such an application of the model as future work.

Figure 4 shows an example of the model's progression of edge beliefs when training on data generated by a linear SCM following a chain graph structure. This illustrates how the edge beliefs start at $\sigma(\gamma_{ij}) = 0.5$ for all $i \neq j$ after which they increase for true edges and decrease—in this case rather rapidly—for non-existent edges. For more examples like this on larger graphs, we refer the reader to Appendix A.4.

5.4.2 Synthetic Nonlinear Data

In order to evaluate our method on nonlinear data, we follow the approach in (11) when setting the value of root nodes. For all non-root nodes X_i, we set

$$X_i := f(X_{\mathbf{PA}_i^{\mathcal{G}^*}}) + 0.4 \cdot U_i,$$

where f is an MLP consisting of one 10-unit hidden layer using a Leaky ReLU nonlinearity with a negative slope of 0.25 and all parameters sampled from $\mathcal{N}(0, 1)$. U_i is again obtained according to (11). This remains in line with the approach taken by Brouillard et al. (2020).

We find that our model has much more difficulties uncovering causal structure when the functional relationships are nonlinear. While the mean SHD is generally below 1 on datasets of $N \leq 4$ variables, predictive accuracy declines sharply for $N \geq 6$. This holds across all graph structures, especially graphs with high edge density. Inspecting the results for full graphs (i.e. graphs with the highest possible edge density), we find that the method suffers from "catastrophic failure" on virtually every run in our experiments. For sparse graphs, the model exhibits much better performance, although the SHD remains higher than that obtained on linear data.

	Structured						Random	
N	Chain	Collider	Tree	Bidiag	Jungle	Full	ER-1	ER-2
2	0 ± 0	-	-	-	-	-	-	-
4	0.2 ± 0.4	0.6 ± 1.2	0.8 ± 0.4	1.0 ± 0.6	0.4 ± 0.8	1.2 ± 1.5	0.2 ± 0.4	-
6	3.4 ± 1.0	1.2 ± 2.4	1.8 ± 1.2	3.6 ± 1.3	4.2 ± 1.7	5.8 ± 4.1	2.6 ± 1.7	5.0 ± 2.4
8	5.4 ± 2.7	3.4 ± 6.8	3.6 ± 1.5	7.4 ± 3.5	6.8 ± 2.7	17.4 ± 5.1	0.8 ± 0.7	12.4 ± 5.2
10	7.6 ± 3.4	4.2 ± 6.4	4.0 ± 2.4	14.2 ± 6.1	9.8 ± 2.7	40.0 ± 2.8	2.8 ± 2.4	15.2 ± 4.3

Table 2: Mean SHD (lower is better) and standard deviation for our proposed method on nonlinear SCMs with structured and random graphs. Each benchmark was obtained by running our method 5 times on SCMs with randomly generated parameters.

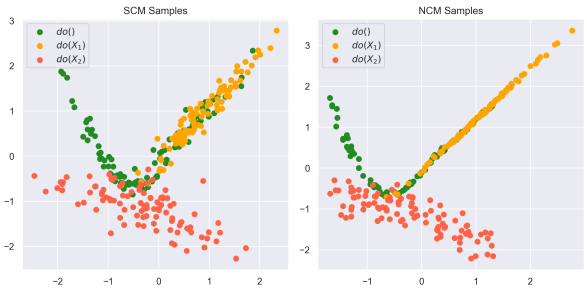


Figure 5: Left: Sampled (X_1, X_3) pairs drawn from a nonlinear SCM with a 3-node bidiagonal causal graph. Different colors represent different interventional distributions. Right: Same as left but with samples drawn from an NCM trained adversarially on data from the SCM.

Interestingly, we also find that we need to regularize the loss function even further in order to obtain useful results on nonlinear data. Specifically, we find that the addition of L2 regularization on the model's MLPs benefits predictive accuracy significantly. Using the same L2 regularization on linear data similarly seems to boost predictive accuracy on sparse graphs. However, accuracy on dense graphs is substantially reduced despite the L2 regularization not being applied on the edge beliefs themselves. This leads us to think that while L2 regularization seems necessary for useful predictions on nonlinear data, it may simultaneously be partly responsible for the model's poor performance on graphs with high edge density. Had we had more time, we would have liked to explore this phenomenon further.

Figure 5 (left) shows a scatter plot of 100 samples drawn from both observational and different interventional distributions of a nonlinear SCM with a 3-node bidiagonal causal graph (i.e. a graph with edges $X_1 \to X_2$, $X_2 \to X_3$, and $X_1 \to X_3$). Figure 5 (right) shows 100 samples drawn from an approximation of the same distributions, modeled by an NCM. For nonlinear data obeying small causal graphs such as this, we see that our method is able to successfully model observational and interventional distributions due to it having 1) predicted the correct causal structure during training and 2) accurately modeled the mechanisms producing the variables in question from the values of their parents.

5.4.3 Effect of Regularization

Similar to other neural causal discovery methods in the literature, we find regularization to be essential for accurate predictions (Kalainathan et al. 2018; Ke et al. 2020; Brouillard et al. 2020). More specifically, we find that it is important to find a suitable configuration of the coefficients λ_E and λ_{DAG}. When λ_E is small, the model's predictions are usually characterized by the inclusion of superfluous edges, whereas when λ_E is high, the model favors sparsity to such an extent that it often fails to include true edges. Similarly, the coefficient λ_{DAG} on the cycle penalty is important for obtaining acyclic edge predictions but can be harmful when applied throughout training. For instance, a true edge can sometimes be "ruled out" prematurely (i.e. by having its associated edge belief set to such a low value that it is rarely sampled again) if the edge at any point violates acyclicity in the graph as a whole. Importantly, this is always the case in the beginning of training, since the initial edge beliefs are set such that $\sigma(\gamma_{ij}) = 0.5$ for all pairs $(i, j) : i \neq j$. For this reason, we have found it helpful to gradually increase the value of λ_{DAG} as described in §5.2. As already mentioned, we also find that decaying the value of λ_E helps the model increase its confidence on true edges in the latter stages of training.

In order to analyze these phenomena in more detail, we conduct an ablation study on λ_E and λ_{DAG} to ascertain the importance of each. We do this in the context of graphs of $N = 5$ nodes. Table 4 shows the results of running our method on data from linear SCMs based on random graphs sampled according to the ER-1 scheme. (Note that "activating" each regularization coefficient corresponds to doing so using the described mechanism for increasing or decaying its value.)

	Nonlinear						Linear	
	Collider	Chain	Jungle	Collider	Full	ER-1	ER-1	
N	7	8	8	8	8	10	10	Data
Zheng et al. 2018	11	24	14	18	21	-	-	D
Y. Yu et al. 2019	6	7	12	7	25	-	-	D
Heinze-Deml et al. (2018)	6	7	12	7	28	-	-	D
Peters, Bühlmann, et al. 2016	4	3	8	2	16	-	-	D
Eaton et al. 2007	7	0	0	7	1	-	-	D
Ke et al. 2020	0	0	0	0	0	-	-	D
Brouillard et al. 2020	-	-	-	-	-	$4.2_{\pm 5.6}$	$0.9_{\pm 1.3}$	C
Ours	$0.6_{\pm 1.2}$	$5.4_{\pm 2.7}$	$6.8_{\pm 2.7}$	$3.4_{\pm 6.8}$	$17.4_{\pm 5.1}$	$2.8_{\pm 2.4}$	$0.4_{\pm 0.5}$	C

Table 3: Benchmarks obtained by other methods in the literature on structured and random graphs using linear and nonlinear data. "D" indicates use of discrete data, while "C" indicates use of continuous data. For more, see Ke et al. 2020 and Brouillard et al. 2020.

Regularization	**Mean SHD**
No Regularization	15.0 ± 0.9
$\lambda_E J_E$	3.4 ± 1.6
$\lambda_{\text{DAG}} J_{\text{DAG}}$	1.2 ± 1.2
$\lambda_E J_E + \lambda_{\text{DAG}} J_{\text{DAG}}$	0 ± 0

Table 4: Mean SHD and standard deviation after running our proposed method on linear datasets generated on the basis of ER-1 graphs with $N = 5$. The results were obtained by running our method 5 times for each configuration.

We find that omitting regularization entirely has a significant negative impact on the performance of our method. Concretely, we find that the model increases its edge beliefs on erroneous edges when sparsity is not enforced. This is made worse by the fact that it does so on *all* edges, given that the cycle penalty is also omitted. In every one of our experiments, $\sigma(\gamma_{ij}) > 0.2$ for all $(i, j) : i \neq j$. In other words, when omitting all regularization, the model consistently predicts a fully connected graph, regardless of the ground truth DAG. As soon as sparsity is enforced, predictions improve significantly. However, we still observe that the model has a tendency to put high edge beliefs on both causal and **anti-causal** edges (i.e. edges going in the opposite direction of the true causal direction). In the absence of J_{DAG}, such predictions are not penalized. We see further improvements still when only penalizing cycles. This is somewhat surprising, given the apparent "eagerness" of the model to include as many edges as possible when no regularization is used. Finally, and perhaps unsurprisingly, we find that the model is able to perfectly identify the causal graphs when both sparsity and DAGness is enforced.

5.4.4 Comparison With Other Methods

We find that comparing our method to other causal discovery techniques in the literature poses somewhat of a challenge. This is, among other things, due to the fact that benchmarks are often performed on graphs of higher cardinality than what we had the time and resources to explore. Furthermore, we find that there is a large variety in the type of data used (categorical or continuous) as well as in datasets (real or synthetic) and type of interventions (hard, soft, known, unknown, or uncertain). Despite this, Table 3 shows a compilation of benchmarks on both linear and nonlinear data found in Ke et al. 2020 and Brouillard et al. 2020. It should be noted that many of these benchmarks are not fully comparable to the ones obtained by our method. For instance, some of them operate exclusively on categorical data and/or rely on soft—and in some cases even unknown—interventions that differ from the hard and known interventions used in this work. Still, it should give an indication of how other causal discovery methods in the literature perform on graphs identical to the ones on which we evaluated our method.

The best comparison is between our method and the method proposed by Brouillard et al. 2020. This is due to the fact that we follow an identical procedure for generating the datasets on which the methods are evaluated. Furthermore, their method similarly operates on continuous data using known, hard interventions. We see that we outperform their method on both linear and nonlinear ER-1 graphs of $N = 10$ nodes. However, it should be noted that their method performs much better on denser graph structures. Concretely, they see a mean SHD of 5.5 on ER-4 graphs with 10 nodes (i.e. random graphs with 40 expected edges). Although we do not evaluate our method on ER-4 graphs specifically, full graphs of

10 nodes have 45 edges and are thus comparable to ER-4. Here, our method gets a mean SHD of 40. This result, in particular, highlights that our method is unsuitable for causal discovery on large graphs with high edge density.

6 Discussion & Future Work

The method we explored has a number of obvious shortcomings as well as a few benefits compared to existing methods in the literature. First and foremost, the method's poor performance on nonlinear data with high edge density in the underlying causal graph makes it unsuitable for causal discovery in such domains. Additionally, the current implementation is not fully optimized for parallelization, which poses a more practical problem for the application of the method, although not an insurmountable one. The benefit of the model is largely due to the simplicity of its architecture. Most current neural causal discovery methods rely on maximum likelihood estimation, which means that, in the case of continuous data, they have to make use of models such as normalizing flows to estimate the conditional densities of each variable. Normalizing flows place hard restrictions on the model architecture as they require all transformations of their input to be invertible and have tractable Jacobians (Kobyzev et al. 2020). Conversely, the implicit nature of adversarial training makes it agnostic to the architecture used for generating the data (insofar as the architecture supports gradient-based optimization). This allows for more flexibility and ease of use but comes at the cost of a more opaque and arguably less principled approach to modeling the data distribution.

The performance of our method, like that of many others in the literature, is highly sensitive to regularization. Much of the work in this project was thus spent trying to identify a proper set of regularization terms as well as their associated hyperparameters. Other efforts included various tweaks to the training procedure, such as:

- Drawing on graph per batch of samples in an effort to stabilize training. Ultimately, we found that drawing one graph per sample resulted in better predictions, likely due to the model seeing a broader range of candidate graphs.
- Intervening on the same variable for all samples in a batch. Again, we found that randomizing interventions across all samples in a batch resulted in higher accuracy.
- Pretraining the MLPs prior to updating the edge beliefs. Our method remains somewhat sensitive to differences in initialization, and pretraining the MLPs was attempted in an effort to alleviate this problem. However, how to do so in the context of self-contained NCMs is not obvious, and our efforts ultimately made little to no difference in terms of predictive accuracy.
- ... and more.

These efforts were complicated by the fact that evaluating our method on a sufficiently large number of graphs to ascertain whether or not the changes constituted an improvement was time-consuming, particularly for larger graphs. Although our method's performance on nonlinear data leaves something to be desired, we believe it is an interesting finding, in and of itself, that this mode of training works *at all*, albeit most convincingly on linear data. Given the method's sensitivity to regularization—which we analyzed in the ablation study in §5.4.3—it should not be ruled out that a minor change in this regard could substantially improve the method's performance on nonlinear data. Ideally, we would have liked to explore this question further. However, due to time constraints, we must leave it as an idea for future work.

A possible critique of our method is that it complicates the optimization procedure unnecessarily by imposing that the NCM be self-contained throughout training. This forces each MLP to approximate the conditional distribution of its associated variable by taking as input values which are themselves not optimized yet. This could potentially explain why our method—despite working for smaller graphs—falls short when data complexity and/or variable cardinality increases. We thus believe it would be interesting to explore how training the model in a non-self-contained fashion would affect its performance. This would put our work more in line with that of Kalainathan et al. 2018, although our use of interventional data would be a point of departure.

We are also intrigued by the idea of incorporating a heuristic for **active interventions** (Scherrer et al. 2021). This would enable the model to actively choose which interventional distribution to optimize, and could, for instance, be done by identifying which interventional distribution produces the highest binary cross-entropy loss during training. An increased number of samples could then be drawn from that distribution so as to improve the model's generative performance and, by extension, its predictive accuracy in terms of causal discovery. Finally, had we had more time, we would have also liked to develop a faster implementation of our algorithm, specifically focusing on speeding up sampling from the NCM. This would likely have enabled us to evaluate our method on larger graphs, which would also have made compar-

isons to other causal discovery methods in the literature easier.

7 Conclusion

In this work, we motivated the need for incorporating causal modeling into machine learning systems. We touched on several concepts fundamental to the study of causal inference, and we introduced the problem of causal discovery. Our primary contribution was to introduce a novel method of score-based causal discovery from interventional and continuous data. Our method relied on a soft adjacency matrix of edge beliefs, representing the model's confidence in the existence of edges in the causal graph underlying the dataset on which the model is trained. We showed how an ability to sample DAGs from our matrix of edge beliefs allows for the construction of what we call self-contained neural causal models (i.e. neural network-based causal models taking as input only a noise vector sampled from a prior distribution), and we explored how such a model might be trained adversarially by introducing a discriminator optimized to distinguish "true" samples from "fake" samples. We demonstrated that training a neural causal model in this fashion can, under suitable assumptions, be used as a method of causal discovery. We found our method to work best on linear data generated by causal graphs of low to medium edge density, while predictive accuracy declined when run on nonlinear data.

Acknowledgements

Prior to beginning this project, I was afraid I would not be able to find a supervisor open to exploring the intersection of machine learning and causality. Many thanks to Jeppe Nørregaard for agreeing to take on that role and for providing me with the freedom to explore the topics closest to my interests. I also thank him for consistently providing useful ideas and encouraging comments at our biweekly meetings at ITU.

References

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25.

Szegedy, Christian et al. (2017). "Inception-v4, inception-resnet and the impact of residual connections on learning". In: *Thirty-first AAAI conference on artificial intelligence*.

Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Advances in neural information processing systems* 30.

Brown, Tom et al. (2020). "Language models are few-shot learners". In: *Advances in neural information processing systems* 33, pp. 1877–1901.

Hinton, Geoffrey et al. (2012). "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal processing magazine* 29.6, pp. 82–97.

He, Yanzhang et al. (2019). "Streaming end-to-end speech recognition for mobile devices". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 6381–6385.

Mnih, Volodymyr et al. (2015). "Human-level control through deep reinforcement learning". In: *nature* 518.7540, pp. 529–533.

Silver, David, Julian Schrittwieser, et al. (2017). "Mastering the game of go without human knowledge". In: *nature* 550.7676, pp. 354–359.

Silver, David, Thomas Hubert, et al. (2017). "Mastering chess and shogi by self-play with a general reinforcement learning algorithm". In: *arXiv preprint arXiv:1712.01815*.

Jumper, John et al. (2021). "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873, pp. 583–589.

Ramesh, Aditya et al. (2022). "Hierarchical text-conditional image generation with clip latents". In: *arXiv preprint arXiv:2204.06125*.

Marcus, Gary and Ernest Davis (2019). *Rebooting AI: Building artificial intelligence we can trust*. Vintage.

Hartnett, Kevin (2018a). "Machine learning confronts the elephant in the room". In: *Quanta Magazine*.

Pearl, Judea (2019). "The seven tools of causal inference, with reflections on machine learning". In: *Communications of the ACM* 62.3, pp. 54–60.

Shorten, Connor and Taghi M Khoshgoftaar (2019). "A survey on image data augmentation for deep learning". In: *Journal of big data* 6.1, pp. 1–48.

Du, Xuefeng et al. (2022). "VOS: Learning What You Don't Know by Virtual Outlier Synthesis". In: *arXiv preprint arXiv:2202.01197*.

Schölkopf, Bernhard et al. (2021). "Towards causal representation learning". In: *arXiv preprint arXiv:2102.11107*.

Popper, Karl (2014). *Conjectures and refutations: The growth of scientific knowledge*. routledge.

Deutsch, David (2012). "How close are we to creating artificial intelligence?" In: *AEON Magazine* 10.3.

– (2011). *The beginning of infinity: Explanations that transform the world*. Penguin UK.

Gopnik, Alison et al. (2004). "A theory of causal learning in children: causal maps and Bayes nets." In: *Psychological review* 111.1, p. 3.

Sloman, Steven, Aron K Barbey, and Jared M Hotaling (2009). "A causal model theory of the meaning of cause, enable, and prevent". In: *Cognitive Science* 33.1, pp. 21–50.

Khemlani, Sangeet S, Aron K Barbey, and Philip N Johnson-Laird (2014). "Causal reasoning with mental models". In: *Frontiers in human neuroscience* 8, p. 849.

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

Hartnett, Kevin (2018b). "To build truly intelligent machines, teach them cause and effect". In: *Quanta Magazine*.

Spirtes, Peter et al. (2000). *Causation, prediction, and search*. MIT press.

Bengio, Yoshua et al. (2019). "A meta-transfer objective for learning to disentangle causal mechanisms". In: *arXiv preprint arXiv:1901.10912*.

Ke, Nan Rosemary et al. (2020). "Dependency Structure Discovery from Interventions". In.

Zhu, Shengyu, Ignavier Ng, and Zhitang Chen (2019). "Causal discovery with reinforcement learning". In: *arXiv preprint arXiv:1906.04477*.

Friedman, Nir et al. (2000). "Using Bayesian networks to analyze expression data". In: *Journal of computational biology* 7.3-4, pp. 601–620.

Robins, James M, Miguel Angel Hernan, and Babette Brumback (2000). *Marginal structural models and causal inference in epidemiology*.

Hicks, John et al. (1980). *Causality in economics*. Australian National University Press.

Jinek, Martin et al. (2012). "A programmable dual-RNA–guided DNA endonuclease in adaptive bacterial immunity". In: *science* 337.6096, pp. 816–821.

Dixit, Atray et al. (2016). "Perturb-Seq: dissecting molecular circuits with scalable single-cell RNA profiling of pooled genetic screens". In: *cell* 167.7, pp. 1853–1866.

Kocaoglu, Murat et al. (2017). "Causalgan: Learning causal implicit generative models with adversarial training". In: *arXiv preprint arXiv:1709.02023*.

Brouillard, Philippe et al. (2020). "Differentiable causal discovery from interventional data". In: *Advances in Neural Information Processing Systems* 33, pp. 21865–21877.

Scherrer, Nino et al. (2021). "Learning neural causal models with active interventions". In: *arXiv preprint arXiv:2109.02429*.

Goodfellow, Ian et al. (2014). "Generative adversarial nets". In: *Advances in neural information processing systems* 27.

Xia, Kevin et al. (2021). "The causal-neural connection: Expressiveness, learnability, and inference". In: *Advances in Neural Information Processing Systems* 34.

Peters, Jonas, Dominik Janzing, and Bernhard Schölkopf (2017). *Elements of causal inference: foundations and learning algorithms*. The MIT Press.

Coricelli, Giorgio and Aldo Rustichini (2010). "Counterfactual thinking and emotions: regret and envy learning". In: *Philosophical Transactions of the Royal society B: Biological sciences* 365.1538, pp. 241–247.

Bareinboim, Elias et al. (2021). *On Pearl's Hierarchy and the Foundations of Causal Inference*. `https://causalai.net/r60.pdf`.

Pearl, Judea and Dana Mackenzie (2018). *The book of why: the new science of cause and effect*. Basic books.

Shimizu, Shohei et al. (2006). "A linear non-Gaussian acyclic model for causal discovery." In: *Journal of Machine Learning Research* 7.10.

Popper, Karl (2005). *The logic of scientific discovery*. Routledge.

Glymour, Madelyn, Judea Pearl, and Nicholas P Jewell (2016). *Causal inference in statistics: A primer*. John Wiley & Sons.

Verma, Thomas S and Judea Pearl (1991). "Equivalence and synthesis of causal models". In: pp. 255–270.

Glymour, Clark, Kun Zhang, and Peter Spirtes (2019). "Review of Causal Discovery Methods Based on Graphical Models". In: *Frontiers in Genetics* 10, p. 524. issn: 1664-8021. doi: `10.3389/fgene.2019.00524`. url: `https://www.frontiersin.org/article/10.3389/fgene.2019.00524`.

Harary, Frank and Edgar M Palmer (2014). *Graphical enumeration*. Elsevier.

Hyttinen, Antti, Frederick Eberhardt, and Matti Järvisalo (2014). "Constraint-based Causal Discovery: Conflict Resolution with Answer Set Programming." In: *UAI*, pp. 340–349.

Richardson, Thomas S (2013). "A discovery algorithm for directed cyclic graphs". In: *arXiv preprint arXiv:1302.3599*.

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

Hauser, Alain and Peter Bühlmann (2012). "Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs". In: *The Journal of Machine Learning Research* 13.1, pp. 2409–2464.

Zheng, Xun et al. (2018). "Dags with no tears: Continuous optimization for structure learning". In: *Advances in Neural Information Processing Systems* 31.

Kalainathan, Diviyan et al. (2018). "Sam: Structural agnostic model, causal discovery and penalized adversarial learning". In.

Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114*.

Arjovsky, Martin and Léon Bottou (2017). "Towards principled methods for training generative adversarial networks". In: *arXiv preprint arXiv:1701.04862*.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein generative adversarial networks". In: *International conference on machine learning*. PMLR, pp. 214–223.

Karras, Tero, Samuli Laine, and Timo Aila (2019). "A style-based generator architecture for generative adversarial networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410.

Grover, Aditya, Manik Dhar, and Stefano Ermon (2018). "Flow-gan: Combining maximum likelihood and adversarial learning in generative models". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1.

Kobyzev, Ivan, Simon JD Prince, and Marcus A Brubaker (2020). "Normalizing flows: An introduction and review of current methods". In: *IEEE transactions on pattern analysis and machine intelligence* 43.11, pp. 3964–3979.

Pawlowski, Nick, Daniel Coelho de Castro, and Ben Glocker (2020). "Deep structural causal models for tractable counterfactual inference". In: *Advances in Neural Information Processing Systems* 33, pp. 857–869.

Hornik, Kurt (1991). "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2, pp. 251–257.

Williams, Ronald J (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3, pp. 229–256.

Jang, Eric, Shixiang Gu, and Ben Poole (2016). "Categorical reparameterization with gumbel-softmax". In: *arXiv preprint arXiv:1611.01144*.

Tsamardinos, Ioannis, Laura E Brown, and Constantin F Aliferis (2006). "The max-min hill-climbing Bayesian network structure learning algorithm". In: *Machine learning* 65.1, pp. 31–78.

Yu, Yue et al. (2019). "DAG-GNN: DAG structure learning with graph neural networks". In: *International Conference on Machine Learning*. PMLR, pp. 7154–7163.

Heinze-Deml, Christina, Jonas Peters, and Nicolai Meinshausen (2018). "Invariant causal prediction for nonlinear models". In: *Journal of Causal Inference* 6.2.

Peters, Jonas, Peter Bühlmann, and Nicolai Meinshausen (2016). "Causal inference by using invariant prediction: identification and confidence intervals". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 78.5, pp. 947–1012.

Eaton, Daniel and Kevin Murphy (2007). "Belief net structure learning from uncertain interventions". In: *J Mach Learn Res* 1, pp. 1–48.

A Appendix

A.1 Training Algorithm

Algorithm 1 contains the pseudocode for the proposed method of training NCMs.

Algorithm 1 Causal Discovery With Adversarial Training. All experiments used batch size $m = 256$ and learning rates $\alpha_\theta = 10^{-3}$, $\alpha_\phi = 10^{-4}$, $\alpha_\gamma = 5 \cdot 10^{-4}$.

Require: γ, matrix of unnormalized edge beliefs. NCM $\mathfrak{N}$ consisting of MLPs $f_{\theta_1}, ..., f_{\theta_N}$ parametrized by θ_i. $\mathfrak{D}$, discriminator parametrized by ϕ. m, the batch size. n_{batch}, number of batches in one epoch. D, dataset generated by SCM across all interventional distributions.

for $l = 1, ..., 100 \cdot N$ or until convergence **do**
  $\lambda_E \leftarrow \max(0.01, 0.1 - (0.1 \cdot \frac{l}{100 \cdot N}))$
  $\lambda_{\text{DAG}} \leftarrow \frac{l}{100 \cdot N}$
  for $1, ..., n_{\text{batch}}$ **do**
    Sample noise vectors $\{\boldsymbol{z}_1, ..., \boldsymbol{z}_m\} \sim P_{\mathcal{Z}}$ where $\boldsymbol{z}_i \in \mathbb{R}^N$ ▷ Train NCM / edge beliefs
    Sample graphs $\{\mathcal{G}^{(1)}, ..., \mathcal{G}^{(m)}\} \overset{\text{DAG}}{\sim} \sigma(\gamma)$
    Sample values $\{x^{(1)}, ..., x^{(m)}\}$ from discrete uniform distribution over $[0, N]$
    for $\forall x^{(i)} \in \{x^{(1)}, ..., x^{(m)}\} : x^{(i)} > 0$ **do**
      Mutilate $\mathcal{G}^{(i)}$ by removing all incoming edges to node $x^{(i)}$
    end for

    if update edge beliefs **then** ▷ Update edge beliefs
      $g_\gamma \leftarrow \nabla[\frac{1}{m} \sum_{i=1}^{m} \log(1 - \mathfrak{D}(\mathfrak{N}^{\mathcal{G}^{(i)}}(\boldsymbol{z}^{(i)}) \mid\mid \boldsymbol{h}^{(i)})) + \lambda_E J_E + \lambda_{\text{DAG}} J_{\text{DAG}}]$
      $\gamma \leftarrow \gamma - \alpha_\gamma \cdot \text{RMSProp}(\gamma, g_\gamma)$ ▷ Learning rate increase omitted for simplicity
    else ▷ Update NCM parameters
      $g_\theta \leftarrow \nabla[\frac{1}{m} \sum_{i=1}^{m} \log(1 - \mathfrak{D}(\mathfrak{N}^{\mathcal{G}^{(i)}}(\boldsymbol{z}^{(i)}) \mid\mid \boldsymbol{h}^{(i)}))]$
      $\theta \leftarrow \theta - \alpha_\theta \cdot \text{RMSProp}(\theta, g_\theta)$
    end if

    Sample noise vectors $\{\boldsymbol{z}^{(1)}, ..., \boldsymbol{z}^{(m)}\} \sim P_{\mathcal{Z}}$ where $\boldsymbol{z}^{(i)} \in \mathbb{R}^N$ ▷ Train discriminator
    Sample graphs $\{\mathcal{G}^{(1)}, ..., \mathcal{G}^{(m)}\} \overset{\text{DAG}}{\sim} \sigma(\gamma)$
    Sample values $\{x^{(1)}, ..., x^{(m)}\}$ from discrete uniform distribution over $[0, N]$
    for $\forall x^{(i)} \in \{x^{(1)}, ..., x^{(m)}\} : x^{(i)} > 0$ **do**
      Mutilate $\mathcal{G}^{(i)}$ by removing all incoming edges to node $x^{(i)}$
    end for
    Draw random samples $\{\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(m)}\}$ from D
    $g_\phi \leftarrow \nabla[\frac{1}{m} \sum_{i=1}^{m} \log \mathfrak{D}(\boldsymbol{x}^{(i)} \mid\mid \boldsymbol{h}_x^{(i)}) + \log(1 - \mathfrak{D}(\mathfrak{N}^{\mathcal{G}^{(i)}}(\boldsymbol{z}^{(i)}) \mid\mid \boldsymbol{h}_y^{(i)}))]$
    $\phi \leftarrow \phi + \alpha_\phi \cdot \text{RMSProp}(\phi, g_\phi)$
  end for
end for

A.2 Erdős–Rényi DAG Sampling

Algorithm 2 contains the pseudocode for generating random DAGs according to the Erdős–Rényi (ER) model.

Algorithm 2 Erdős–Rényi DAG Sampling

Require: N, size of DAG to generate. e, parameter specifying the density of the sampled DAG.

1: Construct $N \times N$ matrix A
2: $k \leftarrow \frac{N \cdot (N-1)}{2}$ ▷ Maximum number of edges in a DAG
3: **assert** $k \geq N \cdot e$ ▷ Expected number of edges cannot exceed maximum
4: $p \leftarrow \frac{N \cdot e}{k}$ ▷ Probability of including edge
5: Set $A_{ij} = 0$ for all $(i, j) : i \geq j$, and set $A_{ij} = p$ for all $(i, j) : i < j$.
6: **for** $i = 1, ..., N$ **do**
7: **for** $j = 1, ..., N$ **do**
8: $A_{ij} = \text{Ber}(A_{ij})$
9: **end for**
10: **end for**
11: **return** A

In its original formulation, the ER model does not guarantee DAGness of the sampled graphs. However, in line 5, we set the matrix of probabilities to be upper triangular with zeros along the diagonal, which enforces DAGness of the sampled graphs. Note that ER-1 sampling is only defined for graphs of $N \geq 3$ nodes, since the expected number of edges in the graph cannot exceed the maximum number of edges in a DAG. Similarly, ER-2 sampling is only defined for graphs of $N \geq 5$ nodes.

A.3 Structured Graphs

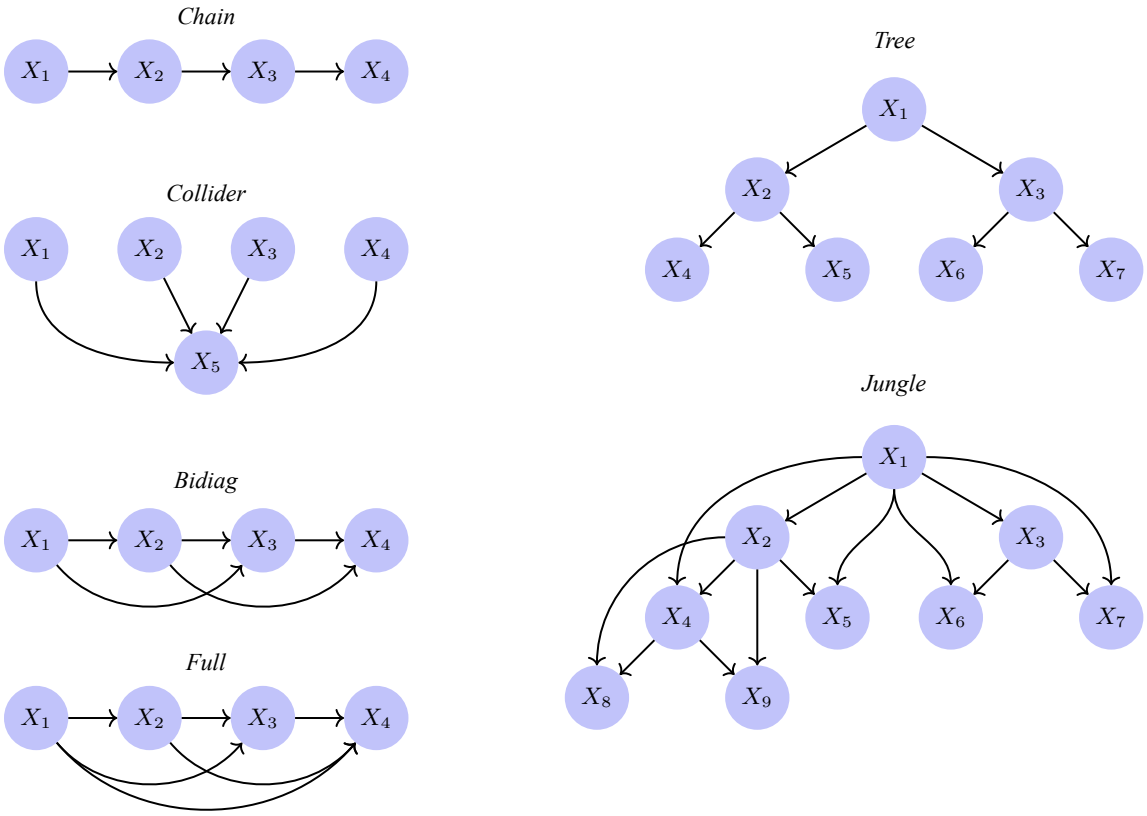


Figure 6

The six structured graphs proposed by Ke et al. (2020) are shown in Figure 6. Chains, colliders and trees have the lowest edge density with $N - 1$ number of edges. Full graphs have the highest edge density with

$$\frac{N \cdot (N-1)}{2}$$

number of edges, the maximum possible in a DAG. The bidiagonal and jungle graphs are ”intermediary” graphs, with an edge density somewhere in between.

A.4 Progression of Edge Beliefs

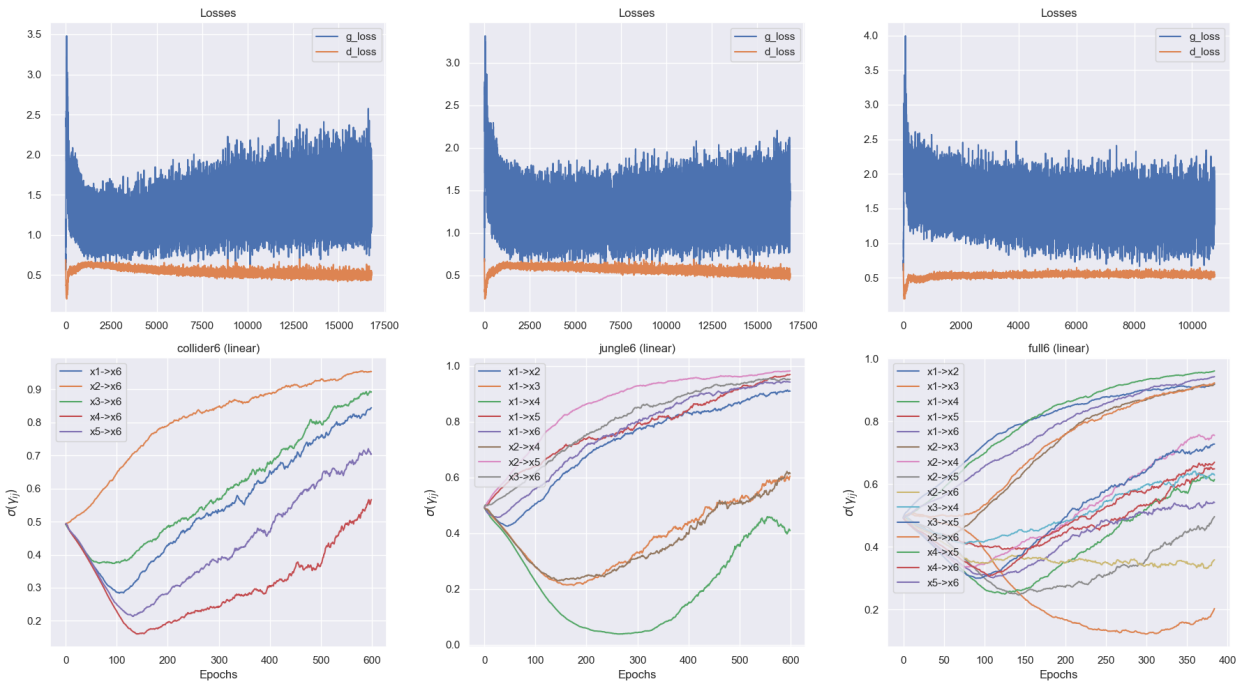


Figure 7: Bottom: Progression of edge beliefs for graphs of different edge densities with $N = 6$. Top: Loss plot showing loss of generator (blue) and discriminator (orange) throughout training.

Figure 7 shows the progression of edge beliefs in our model when trained on a linear dataset of six variables, obeying a collider graph, jungle graph, and full graph, respectively. Each run resulted in an SHD of 0. The plot on the bottom right shows the application of early stopping. Losses for both the generator and discriminator are shown on the top. The generator's loss is consistently higher than the discriminator's due to the added regularization terms.