Towards Modular Learning of Deep Causal Generative Models

Md Musfiqur Rahman [1] **Murat Kocaoglu** [1]

Abstract

Shpitser & Pearl (2008) proposed sound and complete algorithms to compute identifiable observational, interventional, and counterfactual queries for certain causal graph structures. However, these algorithms assume that we can correctly estimate the joint distributions, which is impractical for high-dimensional datasets. During the current rise of foundational models, we have access to large pre-trained models to generate realistic high-dimensional samples. To address the causal inference problem with high dimensional data, we propose a sequential adversarial training algorithm for learning deep causal generative models by dividing the training problem into independent sub-parts, thereby enabling the use of such pre-trained models. Our proposed algorithm called WhatIfGAN, arranges generative models according to a causal graph and trains them to imitate the underlying causal model even with unobserved confounders. Finally, with a semi-synthetic Colored MNIST dataset, we show that WhatIfGAN can sample from identifiable causal queries involving high-dimensional variables.

1. Introduction

Recently there has been an increasing interest in developing causal prediction methods in many applications such as medication effect and recruitment fairness estimation (Castro et al., 2020; Wu et al., 2019). Many existing approaches utilize the structural assumptions encoded in the causal graph to estimate interventional and counterfactual distributions in the presence of unobserved variables, called latent confounders. The well-known identification algorithms (Shpitser & Pearl, 2008) use distributional invariances implied by the causal graph. However, these algorithms require access to the probability distributions of observed variables, except for some special cases (Jung et al., 2020) which is difficult for high-dimensional variables, such as images.

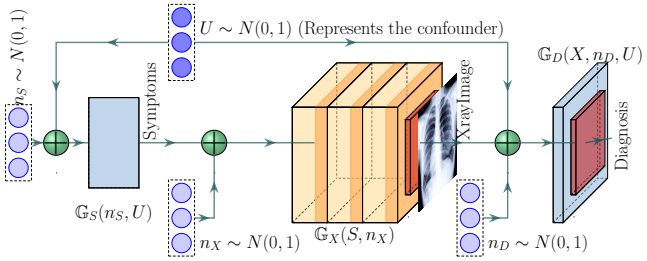


Figure 1. Deep generative model representation for frontdoor graph. Each NN (ex: $\mathbb{G}_S, \mathbb{G}_X, \mathbb{G}_D$) mimics the true mechanism.

Consider the task of predicting a patient's diagnosis from their symptoms given the causal graph: $\text{Symptoms} \rightarrow \text{XrayImage} \rightarrow \text{Diagnosis}, \text{Symptoms} \leftarrow [\text{Hospital Location}] \rightarrow \text{Diagnosis}$. Suppose, we have collected data across two hospitals, but the hospital ID is kept hidden the same as a confounder. The symptoms distribution changes across two hospitals and different methods are also used to classify X-ray images to obtain a diagnosis. This is the front-door graph (Pearl, 2009), where the causal effect of Symptoms on Diagnosis, i.e., $P(\text{Diagnosis}|\text{do}(\text{Symptoms}))$ is uniquely computable or identifiable, with explicit access to the distribution $p(\text{XrayImage}|\text{Symptoms})$. However, we do not have an explicit representation of this high-dimensional image distribution. Thus identification algorithm can not be applied.

In recent years, there has been tremendous improvement in high-dimensional sampling such as generating images and videos with generative adversarial networks (GANs) (Goodfellow et al., 2014; Karras et al., 2019). GANs might require abundant data and resources to train from scratch but if pre-trained, can be used as black boxes. Although they solve high-dimensional sampling, they do not offer any causal capabilities. Kocaoglu et al. (2018) introduces a deep causal model that produces interventional images after training on observational data. However, they assume causal sufficiency, i.e., each observed variable is caused by independent unobserved variables. In contrast, two observable variables

[1]School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana, USA. Correspondence to: Md Musfiqur Rahman <rahman89@purdue.edu>, Murat Kocaoglu <mkocaoglu@purdue.edu>.

Published at the ICML 2023 Workshop on Spurious Correlations, Invariance, and Stability. Honolulu, Hawaii, USA.

sharing an unobserved confounder, i.e., the semi-Markovian model, is quite common in the real world.

For semi-Markovian models, Xia et al. (2021) follows a similar approach as (Kocaoglu et al., 2018) to arrange neural models as a causal graph. For example, the XrayImage causal graph can be represented with generative models as in Figure 1. They propose a minimization-maximization method to identify and estimate causal effects. Xia et al. (2023) extends these to identify and estimate counterfactual queries. However, these methods can not trivially be extended for continuous high-dimensional image data. Besides, they learn semi-Markovian models by training all the variables together, with a common loss function. Training multiple generative models involving both high and low-dimensional variables is challenging. Even if some models are available as pre-trained, these neural causal approaches are not adapted to employ these large generative models as black boxes, failing to utilize their full potential in causality.

In this paper, we propose WhatIfGAN, a modular adversarial learning of deep causal generative models, according to specific sub-graph structures. This modularity allows us to plug in large pre-trained models for specific modules and continue the usual training process for the rest of the graph. In Figure 1, we can plug in a pre-trained GAN model and use its XrayImage output to train Symptoms and Diagnosis mechanisms together. We show that, by using c-component factorization (Tian & Pearl, 2002) and do-calculus rules (Pearl, 1995), we can employ modular training to correctly estimate causal effects such as $P(\text{Diagnosis}|\text{do}(\text{Symptoms}))$ after training converges.

2. Background

Definition 2.1 (Structural causal model, (SCM) (Pearl, 1980))**.** An SCM is a 5-tuple $\mathcal{M} = (G = (\mathcal{V}, \mathcal{E}), \mathcal{N}, \mathcal{U}, \mathcal{F}, P(.))$. $\mathcal{V} = \{V_1, V_2, ..., V_n\}$ is a set of observed random variables. $\mathcal{N}$ is a set of exogenous random variables each causing a single observed variable. $\mathcal{U}$ is a set of latent confounders, i.e., unobserved common causes shared by two observable variables referring to the semi-Markovian causal model. A set of deterministic functions $\mathcal{F}$=$\{f_{V_1}, f_{V_2}, .., f_{V_n}\}$ generates each observed variable from other observed and unobserved variables as $V_i = f_i(Pa_i, E_i, U_{S_i})$, where $Pa_i \subset \mathcal{V}$, $E_i \in \mathcal{N}$ and $U_{S_i} := \{U_j : j \in S_i\}$ for some S_i, such that $U_{S_i} \subset \mathcal{U}$. $\mathcal{P}(.)$ is a product probability distribution over $\mathcal{N}$ and $\mathcal{U}$ and projects a joint distribution $\mathcal{P}_{\mathcal{V}}$ over the variables set $\mathcal{V}$.

An SCM $\mathcal{M}$, induces a causal graph $G = (\mathcal{V}, \mathcal{E})$ containing nodes for each variable $V_i \in \mathcal{V}$. A node V_i is a parent of V_j and represented by a directed edge $V_i \to V_j \in \mathcal{E}$ if only if V_i is in the domain of f_{V_j}. Accordingly, the set $Pa(V_i)$ becomes the parent nodes of the node V_i in the causal graph. G has a bi-directed edge, $V_i \leftrightarrow V_j \in \mathcal{E}$ between V_i and V_j if and only if they share a latent confounder. C-components are maximal set of nodes in G, pairwise connected by bi-directed paths. If there exists a directed path from V_i to V_j then V_i is an ancestor of V_j, i.e., $V_i = An_G(V_j)$. An intervention $\text{do}(v_i)$ replaces f_i with the equation $V_i = v_i$ and in all functions where V_i occurs. The distribution induced on the observed variables after an intervention is $\mathcal{P}_{v_i}(\mathcal{V})$. Graphically, it is represented by $G_{\overline{V_i}}$ where all edges incoming to V_i are removed.

Definition 2.2 (Causal identifiability)**.** Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be disjoint variables sets in G. $P_{\mathbf{x}}(\mathbf{y})$ is defined as the causal effect of the action/intervention $\text{do}(\mathbf{X} = \mathbf{x})$ on the variables in $\mathbf{Y}$. Similarly, $P(\mathbf{Y_x} = \mathbf{y}|e)$ is a counterfactual effect referring to some hypothetical action/intervention $\text{do}(\mathbf{X} = \mathbf{x})$ conditioned on some evidence e, i.e., values of some variables in $\mathcal{V}$. Let $\mathbf{M}$ be the set of all causal models that induce the same causal graph G but possibly different in $\mathcal{N}, \mathcal{U}, \mathcal{F}$. Consider, causal objects $\phi = P(\mathbf{Y_x} = \mathbf{y}|e)$ (or, $P_{\mathbf{x}}(\mathbf{y})$) and $\theta = [P(\mathcal{V}), P(\mathcal{V}|\text{do}(\mathbf{Z'})), \forall \mathbf{Z'} \subseteq \mathbf{Z}]$ (or, $P(\mathcal{V})$) are computable from each model in $\mathbf{M}$. We define that ϕ is θ-identifiable ($\theta \vdash \phi$) in G if there exists a unique deterministic function g_G determined by the graph structure, such that $\phi = g_G(\theta)$ in any $M \in \mathbf{M}$. Shpitser & Pearl (2007) provided a sound and complete algorithm to derive g_G.

Generative adversarial networks (GANs) (Goodfellow et al., 2014). GANs are neural generative models that can sample from high dimensional non-parametric data distributions. A generator network produces samples through a feed-forward computation. Another trainable neural network called the discriminator penalizes the generator by differentiating its output as real or fake. The goal of the generator is to convince the discriminator that the fake samples are from the real data distribution.

3. WhatIFGAN Modular Training

The conditional GAN architecture (Mirza & Osindero, 2014) can be arranged according to a causal graph to produce samples of each variable, by feeding the samples obtained from its parents' generators (Kocaoglu et al., 2018). Besides the independent exogenous noise, for the semi-Markovian model, we feed the same confounding noise samples into the generators of two variables that are confounded in the causal graph. Consider the front-door graph of Figure 1, for the semi-Markovian causal model: $S = f_S(n_S, U), X = f_X(S, n_X), D = f_D(X, n_D, U)$. Here, n_S, n_X, n_D are exogenous noises and U is the confounder between S and D. To learn the structural functions f_S, f_X, f_D of this causal model, we organize a set of neural networks, $\mathbb{G}_S, \mathbb{G}_X, \mathbb{G}_D$ as shown in Figure 1. During the forward pass, we sample Gaussian noise U and feed it into both $\mathbb{G}_S$ and $\mathbb{G}_D$. This architecture ensures that $\mathbb{G}_S$ and $\mathbb{G}_D$ train themselves to learn

the dataset distributions while affecting each other through confounding paths. We utilize the generative models as below:

Definition 3.1 (Deep Causal Generative Model, DCM)**.** Let $D = (\mathcal{V}, \mathcal{E})$ be a causal graph and $\mathbf{Z} = \{Z_1, Z_2, \ldots, Z_n\}$ and $\mathbf{U} = \{U_1, U_2, \ldots, U_m\}$ be two sets of mutually independent random variables to be used in place of unknown exogenous $\mathcal{N}$ and confounding noises $\mathcal{U}$ of the true SCM. Here, $n = |\mathcal{N}|, m = |\mathcal{U}|$. Let T_i index the noise terms that will act as confounders adjacent to V_i in the GAN. $\forall i, j$, If $V_i \leftrightarrow V_j \in \mathcal{E}$ then $|U_{T_i} \cap U_{T_j}| = 1$ and if $V_i \leftrightarrow V_j \notin \mathcal{E}$ then $U_{T_i} \cap U_{T_j} = \emptyset$, i.e., two variables share one noise variable . A collection of feed-forward neural networks $\mathbb{G}$ with the output $\mathbb{G}(Z, U) = [\mathbb{G}_1(Z, U), \mathbb{G}_2(Z, U), \ldots, \mathbb{G}_n(Z, U)]$ is defined as a Deep Causal Generative Model (**DCM**) for the causal graph G if $\exists$ a set of neural network layers f_i such that $\mathbb{G}_i(Z, U) = f_i(\{\mathbb{G}_j(Z, U)\}_{j \in Pa_i}, Z_i, U_{T_i}), \forall i \in [n]$. DCM induces the distribution $Q(.)$. DCM can represent any SCM with the same graph G given sufficiently representative neural nets and large random noises.

3.1. Modular Training

In order to train deep causal generative models in a modular fashion and utilize pre-trained models, we propose a modular training algorithm WhatIfGAN. When there exist unobserved confounders, for example in Figure 2, we can not train each node's mechanism separately by matching only conditional distributions $p(x|pa_x)$ as we can do for causal sufficient case. Suppose, we have an observational dataset $D \sim P(\mathcal{V})$ and our goal is to train a DCM to induce this distribution, i.e., $Q(\mathcal{V}) = P(\mathcal{V})$. If we train the generator $\mathbb{G}_S$ first, and the rest of the mechanisms later, $\mathbb{G}_S$ can match the marginal distributions $P(S)$ but might ignore the dependence induced by the unobserved confounder between S and D since there is no incentive for that. By ignoring the confounders, it is not possible to induce dependence $S \not\perp\!\!\!\perp D|X$. As a result, $Q(S, X, D)$ is not guaranteed to match with $P(S, X, D)$. This suggests that causal mechanisms in the same c-component (here $\mathbb{G}_S$ and $\mathbb{G}_D$) should be trained together with a common loss.

To match the joint distribution $P(\mathcal{V})$ for semi-Markovian models, we propose using Tian's factorization (Tian & Pearl, 2002). It factorizes $P(\mathcal{V})$ into c-factors: the joint distributions of each c-component S_j intervened on their parents, i.e., $P_{pa(S_j)}(S_j)$. For graph G, $P(v) = P(x|\text{do}(s))P(s, d|\text{do}(x))$. This factorization suggests that fitting $P(\mathcal{V})$ is equivalent to fitting each of the c-factors.

Since, we only have access to the $P(\mathcal{V})$ dataset, we utilize $P(\mathcal{V})$ distribution to fit these c-factors by leveraging the do-calculus rule-2 (Pearl, 1995) and modularize the training process. In Figure 2, $P(x|\text{do}(s)) = P(x|s)$ since do-calculus rule-2 applies, i.e., intervening on S is equivalent

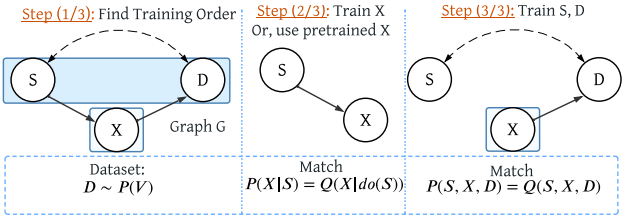


Figure 2. Modular Training on front-door graph

to conditioning on S. We can use the conditional distribution as a proxy to the c-factor to learn $Q(x|\text{do}(s))$ with $\mathbb{G}_X$. However, $P(s, d|\text{do}(x)) \neq P(s, d|x)$. To find a proxy distribution to train the mechanisms in a c-component $\mathbf{Y}$ (here $\{S, D\}$), we search for an ancestor set $\mathcal{A}$ (i.e., $\{X\}$) such that for the parent set $Pa(\mathbf{Y} \cup \mathcal{A})$ (i.e., $\emptyset$), rule-2 applies. Thus, we match $P(s, x, d)$ as proxy to $P(s, d|\text{do}(x))$ and train $\mathbb{G}_S$ and $\mathbb{G}_D$. Ancestor $\mathcal{A}$ satisfies the condition below:

Definition 3.2 (Modularity condition)**.** Given a graph G and a c-component variable set $\mathbf{Y}$, a set $\mathcal{A} \subseteq An(\mathbf{Y}) \setminus \mathbf{Y}$ is said to satisfy the modularity condition if it is the smallest set that satisfies $P(\mathbf{Y} \cup \mathcal{A}|\text{do}(Pa(\mathbf{Y} \cup \mathcal{A}))) = P(\mathbf{Y} \cup \mathcal{A}|Pa(\mathbf{Y} \cup \mathcal{A}))$, i.e., do-calculus rule-2 (Pearl, 1995) applies.

We train the c-components one by one by matching the distribution implied by ancestor set $\mathcal{A}$ that satisfies the modular condition. For each c-component, we train the mechanisms in $\mathbf{Y}$ to learn the c-factor-alternative $P(\mathbf{Y} \cup \mathcal{A}|\text{do}(Pa(\mathbf{Y} \cup \mathcal{A})))$ distribution by matching the observational distribution $P(\mathbf{Y} \cup \mathcal{A}|Pa(\mathbf{Y} \cup \mathcal{A}))$. Since this is sampled from simply $P(\mathcal{V})$ dataset, it does not require $Pa(\mathbf{Y} \cup \mathcal{A})$ to be intervened on. Now, to match $P(\mathbf{Y} \cup \mathcal{A}|Pa(\mathbf{Y} \cup \mathcal{A})) = Q(\mathbf{Y} \cup \mathcal{A}|\text{do}(Pa(\mathbf{Y} \cup \mathcal{A})))$ with our generative models, we pick the values of $Pa(\mathbf{Y} \cup \mathcal{A})$ from $P(\mathcal{V})$ dataset and intervene in our DCM with those values. Since we do not need generated samples for $Pa(\mathbf{Y} \cup \mathcal{A})$ from DCM, we do not require them to be pre-trained. Thus, at step 2/3 in Figure 2, to match $P(x|s) = Q(x|\text{do}(s))$ with our generative models, we pick the observations of S from $P(\mathcal{V})$ dataset and intervene in our DCM without any involvement of $\mathbb{G}_S$.

At step 3/3, we train mechanisms of the next c-component i.e. $[\mathbb{G}_S, \mathbb{G}_D]$. Ancestor set $\mathcal{A} = \{X\}$ satisfies the modularity condition for $\mathbf{Y} = \{S, D\}$. As a proxy to the c-factor of $\mathbf{Y}$, we match $P(\mathbf{Y} \cup \mathcal{A}|\text{do}(Pa(\mathbf{Y} \cup \mathcal{A}))) = P(s, x, d)$. Since we train the mechanisms in $\mathbf{Y}$ only, we need pre-trained $\mathbb{G}_{\mathcal{A}}$ to generate correct samples for the joint. We can either train mechanisms of $\mathcal{A}$ prior to $\mathbf{Y}$ (as we did) or we can use pre-trained mechanisms of $\mathcal{A}$ if provided. In our case, we can freeze training of $[\mathbb{G}_X]$ and use their pre-trained weights from the previous step. We use it only to produce correct samples to feed into $\mathbb{G}_D$ and train $[\mathbb{G}_S, \mathbb{G}_D]$ to match the distribution: $P(s, x, d) = Q(s, x, d)$.

Since $P(s,d|\text{do}(x)) = \frac{P(s,x,d)}{P_s(d)}$ and we have matched both $P(s,x,d)$ and $P_s(d)$, c-factor $P(s,d|\text{do}(x))$ will match as well. However, if we want to match $P(s,x,d)$ directly, we can not use the pre-trained $\mathbb{G}_X$ and it will be hard to reach convergence (please see Section 4 for details).

However, in place of $[\mathbb{G}_X] \rightarrow [\mathbb{G}_S, \mathbb{G}_D]$, if we consider the training order $[\mathbb{G}_S, \mathbb{G}_D] \rightarrow [\mathbb{G}_X]$, it would not be possible to train the mechanisms. Because, to train $[\mathbb{G}_S, \mathbb{G}_D]$ we have to match the joint distribution $P(s,x,d)$ but we do not have $\mathbb{G}_X$ as pre-trained. Thus we always need a valid training order of the c-component mechanisms to match $P(\mathcal{V})$. To obtain a partial order for training the c-components, we construct a directed acyclic graph called $\mathcal{H}$-graph containing c-components as nodes. Unlike $Pa(\mathbf{Y} \cup \mathcal{A})$, conditioning and intervening are not the same for variables in $\mathcal{A}$, i.e., rule-2 does not apply. Thus, to match the alternative distribution $P(\mathbf{Y} \cup \mathcal{A}|Pa(\mathbf{Y} \cup \mathcal{A}))$ while training only $\mathbb{G}_{\mathbf{Y}}$, we need to ensure that the causal mechanisms of $\mathcal{A}$, i.e., $\mathbb{G}_{\mathcal{A}}$ are correctly trained a priori. We express this training order with directed edges from h-nodes containing variables in $\mathcal{A}$ towards the h-node of $\mathbf{Y}$ in the $\mathcal{H}$-graph.

Definition 3.3 ($\mathcal{H}$-graph)**.** Given a causal graph G, let the set of c-components in G be $\mathcal{C} = \{C_1, \dots C_t\}$. Choose a partition $\{H_k\}_k$ of $\mathcal{C}$ such that the $\mathcal{H}$-graph $\mathcal{H} = (V_{\mathcal{H}}, E_{\mathcal{H}})$, defined as follows, is acyclic: $V_{\mathcal{H}} = \{H_k\}_k$ and for any s,t, $H_s \rightarrow H_t \in E_{\mathcal{H}}$, iff $P(H_t|\text{do}(pa(H_t) \cap H_s)) \neq P(H_t|pa(H_t) \cap H_s)$, i.e., do-calculus rule-2 does not hold. Note that one can always choose a partition of $\mathcal{C}$ to ensure $\mathcal{H}$ is acylic: The $\mathcal{H}$ graph with a single node $H_1 = \mathcal{C}$.

For any arbitrary graph (details in Appendix B.2), we run Algorithm 1: **Contruct-H-graph()** to build a $\mathcal{H}$-graph. In Algorithm 2, we train each h-node H_k according to the partial order of $\mathcal{H}$-graph to match with the c-factors-alternative distributions. Finally, Algorithm 6: **TrainModule()** (in Appendix E) employs conditional Wasserstein GAN with penalized gradients (Gulrajani et al., 2017) to train the generators in $\mathbb{G}_{H_k}$ on dataset D such that the joint distributions implied by the set $\mathcal{A}$ matches. This sub-routine uses all mechanisms in $H_k \cup \mathcal{A}$ to produce samples but only updates the mechanisms in $\mathbb{G}_{H_k}$ corresponding to the current h-node. After calling **TrainModule()** for each of the h-nodes according to the partial order of $\mathcal{H}$-graph, WhatIfGAN will find a DCM equivalent to the true SCM that matches $P(\mathcal{V})$ distribution. Thus, it will sample from interventional and counterfactual distributions identifiable from $P(\mathcal{V})$. In Figure 2, we can correctly sample from $P_s(D)$ after training converges. We set $S = s$ instead of using $\mathbb{G}_S$ and propagate those values to generate the rest of the variables. These contributions are presented in Theorem 3.4 and proved in Appendix C.3.

Theorem 3.4. *Let $\mathcal{M}_1 = (G = (\mathcal{V}, \mathcal{E}), \mathcal{N}, \mathcal{U}, \mathcal{F}, P(.))$ be the true SCM and $\mathcal{M}_2 = (G, \mathcal{N}', \mathcal{U}', \mathcal{F}', Q(.))$ be the DCM. Suppose, Algorithm 2, **Modular Training** on Dataset $D \sim P(\mathcal{V})$ converges for each h-node in the $\mathcal{H}$-graph and DCM induces $Q(\mathcal{V})$. Then, we have $i) P(\mathcal{V}) = Q(\mathcal{V})$, and $ii)$ for any interventional or counterfactual causal query $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V})$ identifiable from D, we have $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V}) = \mathcal{K}_{\mathcal{M}2}(\mathcal{V})$.*

4. Experimental Evaluation

To illustrate WhatIfGAN performance in learning the joint distribution involving both low and high-dimensional Colored-MNIST (LeCun et al., 1998) variables, we ran an experiment on the front-door graph $D \rightarrow Image \rightarrow A$, $D \leftrightarrow A$. We constructed a synthetic SCM where a variable U affects both D and A binary variables but is kept hidden in the dataset to make it act like a confounder. Image variable I shows the digit value of D, and A is some attribute of I. Suppose we are given a dataset sampled from $P(D,A,I)$ distribution. Our goal is to estimate the causal effect of D on variable A. To measure the ground truth causal effect, we can use the backdoor criterion (Pearl, 1993), $P(A|do(D)) = \sum_U P(A|D,U)P(U)$ since we have access to U in the true SCM. In the observational dataset, $P(A|do(D))$ is identifiable with the front-door criterion (Pearl, 2009): $P(A|do(D)) = \sum_I P(I|D) \sum_{D'} P(A|D',I)P(D')$. We describe the experimental setup in more detail in Appendix D.2 and provide our implementation at `https://github.com/Musfiqshohan/WhatIfGAN`.

To produce correct samples from $P(A|do(D))$, we have to match $P(D,A,I)$ with the DCM. GAN convergence using this joint distribution loss is difficult since the losses generated by low and high-dimensional variables are not easily comparable and non-trivial to correctly re-weight. Existing causal effect estimation algorithms can not address this since there is no estimator that does not contain explicit image distribution, which is practically impossible to estimate. Thus, we map samples of I to a low-dimensional representation, RI with a trained encoder and match $P(D,A,RI)$ instead of the joint $P(D,A,I)$. We use a fully connected NN to produce D, a deep convolution GAN to generate images, and a classifier to classify images into variable A such that D and A are confounded. For this graph, the corresponding $\mathcal{H}$-graph is $[I] \rightarrow [D,A]$. We can either train $\mathbb{G}_I$ by matching $P(I|D)$ or employ a pre-trained generative model that takes digits D as input and outputs a colored D-digit image. Next, we train $\mathbb{G}_D$ and $\mathbb{G}_A$, matching $P(D,A,I)$ (i.e., $P(D,A,RI)$) since ancestor set is $\mathcal{A} = \{I\}$ for c-component $\{D,A\}$.

In Figure 3, we compare our method with (Xia et al., 2023): NCM and a version of our method: WhatIfGAN-rep that does not use modular training. Since NCM trains all mechanisms with the same loss function calculated from both low and high-dimensional samples, it learns marginal distribution $P(I)$ (Figure 3 col-1) but does not converge to match

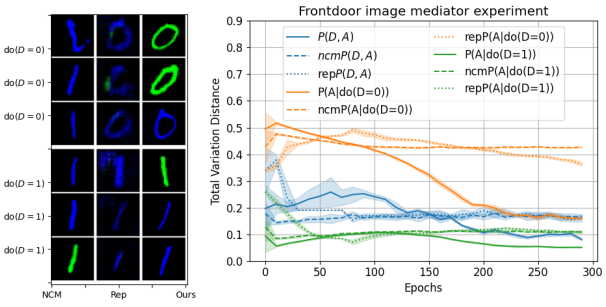


Figure 3. Training on front-door graph: $D \to I \to A$, $D \leftrightarrow A$. Image generation (left) and distribution convergence (right).

$P(D, A, I)$ (dashed-lines). WhatIfGAN-rep uses a low-dim representation of images: RI and matches $P(D, A, RI)$ without modularization. We observe WhatIfGAN-rep to converge (dotted lines) slower compared to the modular training and produce low-quality images (Figure 3 col-2). Finally, WhatIfGAN modular training matches $P(D, A, RI)$ and converges faster (solid-lines) for $P(D, A), P(A|do(D))$ and produces high-quality images from $P(I|do(D))$ (Figure 3 col-3).

5. Conclusion

Our proposed modular training algorithm learns deep causal generative models with high-dimensional variables in the presence of unobserved confounders. After convergence, WhatIfGAN can generate high-dimensional samples from identifiable interventional and counterfactual distributions.

6. Limitations and Future work

Similar to most causal inference algorithms, we had to make the assumption of having a fully specified causal graph with latents, as prior. With the advancements in causal discovery with latents, it might be possible to reliably learn part of the structure and leverage the partial identifiability results from the literature. Indeed, this would be one of the future directions we are interested in. Another limitation of this work is that we assume each confounder to cause only two observed variables which is considered as semi-Markovian in literature. We aim to extend our work for non-Markovian causal models where confounders can cause any number of observed variables.

Acknowledgements

This research has been supported in part by NSF Grant CAREER 2239375.

References

Bareinboim, E. and Pearl, J. Causal inference by surrogate experiments: Z-identifiability. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, UAI'12, pp. 113–120, Arlington, Virginia, USA, 2012. AUAI Press. ISBN 9780974903989.

Castro, D. C., Walker, I., and Glocker, B. Causality matters in medical imaging. *Nature Communications*, 11(1):3673, 2020.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. Generative adversarial nets. In *NIPS*, 2014.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

Jung, Y., Tian, J., and Bareinboim, E. Learning causal effects via weighted empirical risk minimization. *Advances in neural information processing systems*, 33: 12697–12709, 2020.

Karras, T., Laine, S., and Aila, T. 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, 2019.

Kocaoglu, M., Snyder, C., Dimakis, A. G., and Vishwanath, S. Causalgan: Learning causal implicit generative models with adversarial training. In *International Conference on Learning Representations*, 2018.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Mirza, M. and Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Pearl, J. Causality: models, reasoning, and inference, 1980.

Pearl, J. [bayesian analysis in expert systems]: comment: graphical models, causality and intervention. *Statistical Science*, 8(3):266–269, 1993.

Pearl, J. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995.

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

Shpitser, I. and Pearl, J. What counterfactuals can be tested. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pp. 352–359, 2007.

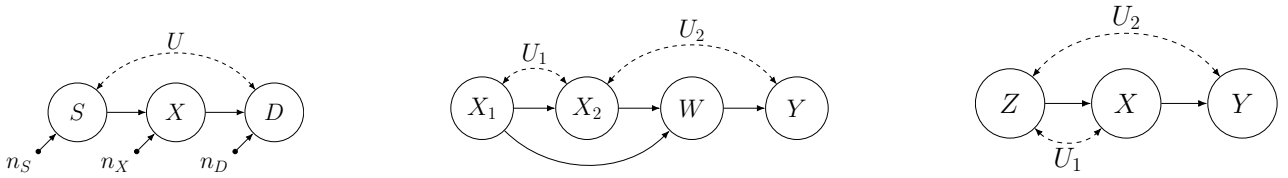
Shpitser, I. and Pearl, J. Complete identification methods for the causal hierarchy. *Journal of Machine Learning Research*, 9:1941–1979, 2008.

Tian, J. and Pearl, J. *A general identification condition for causal effects*. eScholarship, University of California, 2002.

Wu, Y., Zhang, L., Wu, X., and Tong, H. Pc-fairness: A unified framework for measuring causality-based fairness. *Advances in neural information processing systems*, 32, 2019.

Xia, K., Lee, K.-Z., Bengio, Y., and Bareinboim, E. The causal-neural connection: Expressiveness, learnability, and inference. *Advances in Neural Information Processing Systems*, 34:10823–10836, 2021.

Xia, K. M., Pan, Y., and Bareinboim, E. Neural causal models for counterfactual identification and estimation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vouQcZS8KfW.



(a) Front-door graph (b) $\{P(\mathcal{V}), P_{x_1}(\mathcal{V})\} \vdash P_{x_1,x_2}(Y|x_1', x_2')$ (c) $\{P(\mathcal{V}), P_z(X,Y)\} \vdash P_x(Y)$

Figure 4. Causal graphs with latents and respective identifiable causal queries. θ identifies ϕ :$\theta \vdash \phi$

A. Interventional and Counterfactual Sampling

Definition A.1 (Identifiability (Shpitser & Pearl, 2007))**.** Given a causal graph, G, let $\mathbf{M}$ be the set of all causal models that induce G and objects ϕ and θ are computable from each model in $\mathbf{M}$. We define that ϕ is θ-identifiable in G, if there exists a deterministic function g_G determined by the graph structure, such that ϕ can be uniquely computable as $\phi = g_G(\theta)$ in any $M \in \mathbf{M}$.

Definition A.2 (Causal Effects z-Identifiability)**.** Let **X,Y,Z** be disjoint sets of variables in the causal graph G. If $\phi = P_{\mathbf{x}}(\mathbf{y})$ is the causal effect of the action do(**X=x**) on the variables in $\mathbf{Y}$, and θ contains $P(\mathbf{V})$ and interventional distributions $P(\mathbf{V} \setminus \mathbf{Z}'|\text{do}(\mathbf{Z}'))$, for all $\mathbf{Z}' \subseteq \mathbf{Z}$, where ϕ and θ satisfies the definition of Identifiability, we define it as z-identifiabililty. (Bareinboim & Pearl, 2012) proposes a z-identification algorithm to derive g_G for these ϕ and θ

Definition A.3 ($\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$)**.** We represent observational, interventional and counterfactual datasets (queries, distributions) by $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$ datasets (queries, distributions), respectively.

Similar to (Kocaoglu et al., 2018; Xia et al., 2023), we show that a trained DCM can sample from identifiable causal queries from any causal layer. Here we consider $\mathcal{M}_1$ as the true SCM and $\mathcal{M}_2$ as the DCM. $P(.)$ and $Q(.)$ are the probability distributions induced by $\mathcal{M}_1$ and $\mathcal{M}_2$. respectively.

Theorem A.4. *Let $\mathcal{M}_1 = (G = (\mathcal{V}, \mathcal{E}), \mathcal{N}, \mathcal{U}, \mathcal{F}, P(.))$ be an SCM. If a causal query $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V})$ is identifiable from a collection of observational $P(\mathcal{V})$ for graph G, then any SCM $\mathcal{M}_2 = (G, \mathcal{N}', \mathcal{U}', \mathcal{F}', Q(.))$ entails the same answer to the causal query if it entails the same input distributions. Therefore, for any identifiable query $\mathcal{K}$, if $P(\mathcal{V}) \vdash \mathcal{K}_{\mathcal{M}_1}(\mathcal{V})$ and $P(\mathcal{V}) = Q(\mathcal{V})$, then $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V}) = \mathcal{K}_{\mathcal{M}_2}(\mathcal{V})$.*

Proof. By definition of identifiability, we have that $\mathcal{K}_{\mathcal{M}_1} = g_G(P(\mathcal{V}))$ for some deterministic function g_G that is determined by the graph structure. Since $\mathcal{M}_2$ has the same causal graph, the query $\mathcal{K}_{\mathcal{M}_2}$ is also identifiable and through the same function g_G, i.e., $\mathcal{K}_{\mathcal{M}_2} = g_G(Q(\mathcal{V}))$. Thus, the query has the same answer in both SCMs, if they entail the same input distributions over the observed variables, i.e., $P(\mathcal{V}) = Q(\mathcal{V})$. □

Consider the causal graph in Figure 4(b). According to the identification algorithm, the interventional query $\phi = P_{x_1,x_2}(W)$ is identifiable from $\theta = \{P(\mathcal{V})\}$. Let us assume that after training on θ observational data, the joint distribution matches i.e., $P(\mathcal{V}) = Q(\mathcal{V})$. Theorem A.4 implies that $P_{x_1,x_2}(W) = Q_{x_1,x_2}(W)$ holds, i.e., WhatIfGAN will produce correct interventional samples from $P_{x_1,x_2}(W)$ along with other $\mathcal{L}_2$ queries identifiable from $P(\mathcal{V})$. Similarly, in Figure 4(c), the interventional query $\phi = P_x(Y)$ is an unique function of $\theta = \{P(\mathcal{V}), P_z(X,Y)\}$ (Bareinboim & Pearl, 2012). Therefore, after training on datasets from θ distributions, WhatIfGAN will produce correct interventional samples from $P_x(Y)$ and other $\mathcal{L}_2$ queries identifiable from $\{P(\mathcal{V}), P_z(X,Y)\}$. Finally, in Figure 4(b), the counterfactual query $\phi = P_{x_1,x_2}(Y|x_1', x_2')$ is identifiable from $\theta = \{P(\mathcal{V}), P_{x_1}(\mathcal{V})\}$. Thus after training on datasets from θ distributions, WhatIfGAN will produce correct counterfactual samples from $P_{x_1,x_2}(Y|x_1', x_2')$ and other queries identifiable from $\{P(\mathcal{V}), P_{x_1}(\mathcal{V}))\}$.

B. WhatIfGAN: Modular Training for Arbitrary Causal Graphs

In order to train deep causal generative models in a modular fashion for a given causal graph and utilize pre-trained models, we propose a modular training algorithm WhatIfGAN. First, consider a causally sufficient system i.e., no confounder exists. We can train each node's mechanism separately conditioning on its parent nodes (Kocaoglu et al., 2018) since the causal mechanism $p(x|pa_x)$ is fully observed with all its inputs and the output. Then the problem of learning causal

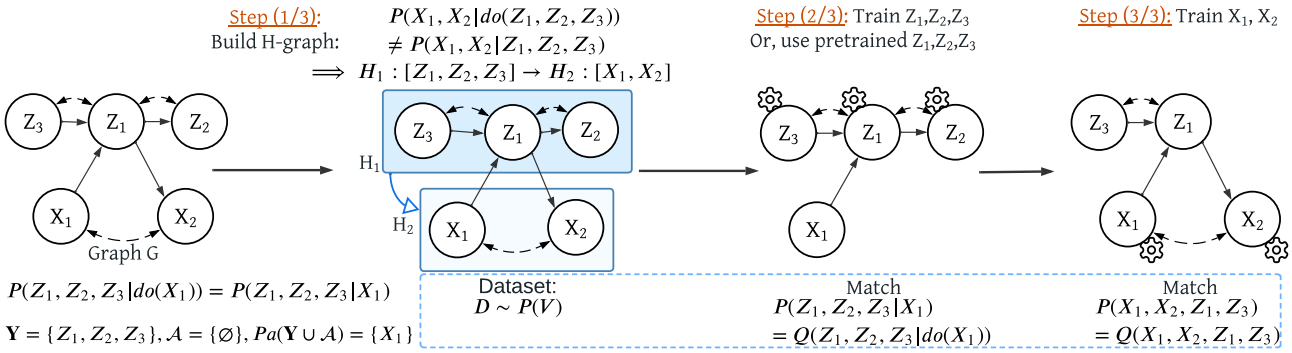


Figure 5. Modular training on H-graph: $H_1 : [Z_1, Z_2, Z_3] \rightarrow H_2 : [X_1, X_2]$ with dataset $D \sim P(V)$.

mechanisms can be solved by learning these conditional distributions. However, this is not the case when some parents become unobserved confounders in the system.

B.1. Basics of Modular Training with Unobserved Confounders

Consider the graph G in Figure 5. Suppose, we have an observational dataset $D \sim P(\mathcal{V})$ and our goal is to train a DCM to induce this distribution, i.e., $Q(\mathcal{V}) = P(\mathcal{V})$. If we train the causal generative model $\mathbb{G}_{Z_1}$ first, and the rest of the mechanisms later, $\mathbb{G}_{Z_1}$ can match the conditional distributions $P(Z_1|X_1, Z_3)$ but might ignore the dependence induced by the unobserved confounders between Z_1, Z_3 or Z_1, Z_2. The training does not provide any incentive for the model to induce the correct dependency. By ignoring the latent confounders, it is not possible to induce dependence between Z_3 and Z_2 conditioned on Z_1. As a result, $Q(Z_1, Z_2, Z_3)$ is not guaranteed to match with $P(Z_1, Z_2, Z_3)$. This simple observation suggests that the causal mechanisms of variables that are in the same c-component should be trained together. Therefore, we have to train $[\mathbb{G}_{Z_1}, \mathbb{G}_{Z_2}, \mathbb{G}_{Z_3}]$ together. For similar reason, we must train $[\mathbb{G}_{X_1}, \mathbb{G}_{X_2}]$ together. This can be accomplished by updating the weights of all neural mechanisms in the same c-component with a common loss.

To match the joint distribution $P(\mathcal{V})$ for semi-Markovian models while preserving the integrity of c-components, we propose using Tian's factorization (Tian & Pearl, 2002). It factorizes $P(\mathcal{V})$ into c-factors: the joint distributions of each c-component S_j intervened on their parents, i.e., $P_{pa(S_j)}(S_j)$.

$$P(v) = P(x_1, x_2|\text{do}(z_1))P(z_1, z_2, z_3|\text{do}(x_1)) \tag{1}$$

This factorization suggests that fitting $P(\mathcal{V})$ is equivalent to fitting each of the c-factors. If we had access to the $\mathcal{L}_2$ distributions $\text{do}(z_1)$ and $\text{do}(x_1), \forall z_1, x_1$, we could intervene on $\mathbb{G}_{Z_1}$ and $\mathbb{G}_{X_1}$ in the DCM to obtain $\text{do}(z_1)$ and $\text{do}(x_1)$ samples. We could train the generative models to match these interventional distributions, which will ensure $P(v) = Q(v)$ in Equation 1 when $P(x_1, x_2|\text{do}(z_1)) = Q(x_1, x_2|\text{do}(z_1))$ and $P(z_1, z_2, z_3|\text{do}(x_1)) = Q(z_1, z_2, z_3|\text{do}(x_1))$.

However, we only have access to the $P(\mathcal{V})$ dataset. Our key idea is to leverage the do-calculus rule-2 (Pearl, 1995) and relate $\mathcal{L}_1$ distribution to $\mathcal{L}_2$ to fit these c-factors and modularize the training process. In Figure 5, $P(z_1, z_2, z_3|\text{do}(x_1)) = P(z_1, z_2, z_3|x_1)$ since do-calculus rule-2 applies, i.e., intervening on X_1 is equivalent to conditioning on X_1. We can then use the conditional distribution as a proxy to the c-factor to learn $Q(z_1, z_2, z_3|\text{do}(x_1))$ with the DCM. However, $P(x_1, x_2|\text{do}(z_1)) \neq P(x_1, x_2|z_1)$. Thus, we include Z_1 into the joint distribution that needs to be fit together with X_1, X_2 and check if the parent set of $\{X_1, X_2, Z_1\}$ satisfy rule-2. We continue until we reach the joint $P(x_1, x_2, z_1, z_3)$ to be the alternative distribution for $\{X_1, X_2\}$'s c-factor. Thus, to find a proxy distribution to train the mechanisms in a c-component $\mathbf{Y}$, we search for an ancestor set $\mathcal{A}$ such that the parent set $Pa(\mathbf{Y} \cup \mathcal{A})$ satisfies rule-2 for the joint. We include the ancestors since only they can affect $\mathbf{Y}$'s mechanisms from outside of the c-component. We restate the **modularity condition** in Definition 3.2 here.

Definition B.1 (Modularity condition)**.** Given causal graph G and a c-component variable set $\mathbf{Y}$, a set $\mathcal{A} \subseteq An(\mathbf{Y})\backslash\mathbf{Y}$ is said to satisfy the modularity condition if it is the smallest set that satisfies $P(\mathbf{Y} \cup \mathcal{A}|\text{do}(Pa(\mathbf{Y} \cup \mathcal{A}))) = P(\mathbf{Y} \cup \mathcal{A}|Pa(\mathbf{Y} \cup \mathcal{A}))$, i.e., do-calculus rule-2 (Pearl, 1995) applies.

To fit the joint distribution according to (1), we identify the modularity condition for each c-component and train them one

by one. For each c-component, we train the mechanisms in $\mathbf{Y}$ to learn the c-factor-alternative $P(\mathbf{Y} \cup \mathcal{A}|\text{do}(Pa(\mathbf{Y} \cup \mathcal{A})))$ distribution by matching the observational distribution $P(\mathbf{Y} \cup \mathcal{A}|Pa(\mathbf{Y} \cup \mathcal{A}))$. Since this is sampled from simply $\mathcal{L}_1$ dataset, it does not require $Pa(\mathbf{Y} \cup \mathcal{A})$ to be intervened on. Now, to match $P(\mathbf{Y} \cup \mathcal{A}|Pa(\mathbf{Y} \cup \mathcal{A}))$ = $Q(\mathbf{Y} \cup \mathcal{A}|\text{do}(Pa(\mathbf{Y} \cup \mathcal{A})))$ with our generative models, we pick the observations of $Pa(\mathbf{Y} \cup \mathcal{A})$ from $\mathcal{L}_1$ dataset and intervene in our DCM with those values. Since we do not need generated samples for $Pa(\mathbf{Y} \cup \mathcal{A})$ from DCM, rather their observations from the given dataset, we do not require them to be trained beforehand. However, the order in which we train c-components matters. For example, G has two c-components in Figure 5. First, consider training the c-components in the $[\mathbb{G}_{Z_1}, \mathbb{G}_{Z_2}, \mathbb{G}_{Z_3}] \rightarrow [\mathbb{G}_{X_1}, \mathbb{G}_{X_2}]$ order. $\mathcal{A} = \emptyset$ satisfies modularity condition for the c-component $\mathbf{Y} = \{Z_1, Z_2, Z_3\}$ since $P(z_1, z_2, z_3|\text{do}(x_1)) = P(z_1, z_2, z_3|x_1)$ holds. At step (2/3) in Figure 5, we can produce samples from the mechanisms of Z_1, Z_2, Z_3 by intervening on their parent X_1 with real observations from dataset D. Thus, we do not need $\mathbb{G}_{X_1}$ to be pre-trained. $[\mathbb{G}_{Z_1}, \mathbb{G}_{Z_2}, \mathbb{G}_{Z_3}]$ converges by matching $P(z_1, z_2, z_3|x_1) = Q(z_1, z_2, z_3|\text{do}(x_1))$.

Now, we train mechanisms of the next c-component $[\mathbb{G}_{X_1}, \mathbb{G}_{X_2}]$ in our training order (step 3/3). Ancestor set $\mathcal{A} = \{Z_1, Z_3\}$ satisfies the modularity condition for $\mathbf{Y} = \{X_1, X_2\}$. As an alternative to the c-factor of $\mathbf{Y}$, we have to match $P(\mathbf{Y} \cup \mathcal{A}|\text{do}(Pa(\mathbf{Y} \cup \mathcal{A}))) = P(x_1, x_2, z_1, z_3)$. But we would like to train the mechanisms in the c-component of $\mathbf{Y}$ only. Therefore, we need $\mathbb{G}_{\mathcal{A}}$ to generate correct samples for the joint. We can either train mechanisms of $\mathcal{A}$ prior to $\mathbf{Y}$ (as we did) or we can use pre-trained mechanisms of $\mathcal{A}$ if provided. In our case, since they were already trained and will never be updated again, we can freeze training of $[\mathbb{G}_{Z_1}, \mathbb{G}_{Z_3}]$ and use their pre-trained weights from the previous step. We use them only to produce correct samples to feed into $\mathbb{G}_{X_2}$ and train $[\mathbb{G}_{X_1}, \mathbb{G}_{X_2}]$ to match the distribution: $P(x_1, x_2, z_1, z_3) = Q(x_1, x_2, z_1, z_3)$. The c-factors in Equation 1, are identifiable from the alternative distributions that we matched above. Therefore, DCM matches the joint distribution $P(V)$ as well.

If we consider the opposite training order of the c-components, i.e., $[\mathbb{G}_{X_1}, \mathbb{G}_{X_2}] \rightarrow [\mathbb{G}_{Z_1}, \mathbb{G}_{Z_2}, \mathbb{G}_{Z_3}]$, it would not be possible to train the mechanisms in them. Because, to train $[\mathbb{G}_{X_1}, \mathbb{G}_{X_2}]$ we have to match the joint distribution $P(x_1, x_2, z_1, z_3)$ as mentioned above. But, we do not have $[\mathbb{G}_{Z_1}, \mathbb{G}_{Z_3}]$ as pre-trained which can produce correct samples. Thus we always need a valid training order of the c-component mechanisms to match $P(\mathcal{V})$. We now have two challenges: to find a valid training order of the c-components and for each c-component, find an ancestor set $\mathcal{A}$ to match the c-factor alternative distributions utilizing the already trained models.

B.2. Algorithm Description

To obtain a partial order for training the c-components, we construct a directed graph structure called $\mathcal{H}$-graph containing c-components as nodes. Suppose, an ancestor set $\mathcal{A}$ satisfies the modularity condition for $\mathbf{Y}$. Even though we will train only $\mathbb{G}_{\mathbf{Y}}$, $\mathcal{A}$ appears together with $\mathbf{Y}$ in the joint distribution that we need to fit. Thus, we need to ensure that the causal mechanisms of $\mathcal{A}$, i.e., $\mathbb{G}_{\mathcal{A}}$ is correctly trained a priori. Therefore, $\mathcal{A}$ should be trained before $\mathbf{Y}$ in the correct training order. On a different note, To construct $\mathcal{A}$, we start with an empty set and check if $Pa(\mathbf{Y} \cup \mathcal{A})$ satisfies the conditions of rule-2 for $\mathbf{Y} \cup \mathcal{A}$. If not, we add parents of $\mathbf{Y} \cup \mathcal{A}$ to construct the new $\mathcal{A}$. We continue the process until $Pa(\mathbf{Y} \cup \mathcal{A})$ satisfies conditions of rule-2. Thus if a c-component contains parents of $\mathbf{Y}$ which does not satisfy the rule-2 for $\mathbf{Y}$, they will eventually be included in $\mathcal{A}$, and thus their c-component has to be trained earlier than $\mathbf{Y}$. We express this training order with a directed edge towards the h-node of $\mathbf{Y}$ in the $\mathcal{H}$-graph. Adding edges in this way will ensure that all variables in $\mathcal{A}$ appear in ancestor h-nodes of $\mathbf{Y}$ in $\mathcal{H}$. While adding edges, we merge c-components on any cycle into a single h-node. Thus some h-node might contain more than one c-component. The final structure is a directed acyclic graph (DAG) and contains a valid partial order $\mathcal{T}$ for modular training. We restate $\mathcal{H}$-graph in Definition 3.3 here.

Definition B.2 ($\mathcal{H}$-graph)**.** Given a causal graph G, let the set of c-components in G be $\mathcal{C} = \{C_1, \dots C_t\}$. Choose a partition $\{H_k\}_k$ of $\mathcal{C}$ such that the $\mathcal{H}$-graph $\mathcal{H} = (V_{\mathcal{H}}, E_{\mathcal{H}})$, defined as follows, is acyclic: $V_{\mathcal{H}} = \{H_k\}_k$ and for any s, t, $H_s \rightarrow H_t \in E_{\mathcal{H}}$, iff $P(H_t|\text{do}(pa(H_t) \cap H_s)) \neq P(H_t|pa(H_t) \cap H_s)$, i.e., do-calculus rule-2 does not hold. Note that one can always choose a partition of $\mathcal{C}$ to ensure $\mathcal{H}$ is acylic: The $\mathcal{H}$ graph with a single node $H_1 = \mathcal{C}$.

We run the subroutine **Contruct-H-graph()** in Algorithm 1 to build $\mathcal{H}$-graphs. We check the edge condition and merge cycles if any. In Figure 5 step (1/3), we build the $\mathcal{H}$-graph $H_1 : [Z_1, Z_2, Z_3] \rightarrow H_2 : [X_1, X_2]$ for G. After constructing the $\mathcal{H}$-graph, in Algorithm 2, we train each h-node H_k according to the partial order $\mathcal{T}$ to match with c-factors-alternative distribution that corresponds to the c-components in H_k. We initialize a set $\mathcal{A} = \{V : V \subseteq An_G(H_k)\}$ to keep track of the joint distribution we need to match to train each h-node H_k. We search for the smallest set of ancestors $\mathcal{A}$, of current h-node H_k such that $\mathcal{A}$ satisfies the modularity condition for H_k tested by Algorithm 7: $\text{IsRule2}(.)$. $\mathcal{A}$ implies the joint distribution

Algorithm 1 Construct_Hgraph(G)

Input: Causal Graph G
$\mathcal{C} \leftarrow$ get_ccomponents(G)
Create nodes $H_k = C_k$ in $\mathcal{H}$, $\forall C_k \in \mathcal{C}$
for each $H_s, H_t \in \mathcal{H}$ such that $s \neq t$ **do**
  if $\exists (V_s, V_t) \in (H_s, H_t)$ such that $V_s = Pa(V_t)$ and $P(H_t|\text{do}(V_s)) \neq P(H_t|V_s)$ **then**
    $\mathcal{H}.add(H_s \rightarrow H_t)$
  end if
end for
$\mathcal{H} \leftarrow$ Merge($\mathcal{H}$, cyc), $\forall cyc \in Cycles(\mathcal{H})$
Return: $\mathcal{H}$

Algorithm 2 Modular Training($G, \mathbf{D}$)

Input: Causal Graph G, Dataset $\mathbf{D}$.
Initialize DCM $\mathbb{G}$
$\mathcal{H} \leftarrow$ Construct_Hgraph(G)
for each $H_k \in \mathcal{H}$ in partial order **do**
  Initialize $\mathcal{A} \leftarrow \mathcal{V}$
  for each $S \subseteq An_G(H_k)$ **do**
    if IsRule2$(H_k, S) = 1 \& |S| < |\mathcal{A}|$ **then**
      $\mathcal{A} \leftarrow S$
    end if
  end for
  $\mathbb{G}_{H_k} \leftarrow$TrainModule($\mathbb{G}_{H_k}, G, H_k, \mathcal{A}, \mathbf{D}$)
end for
Return: $\mathbb{G}$

in Equation 2, which is sufficient for training the current h-node H_k to learn the c-factors $P_{Pa(C_i)}(C_i), \forall C_i \in H_k$.

$$Q(H_k \cup \mathcal{A}|\text{do}(pa(H_k \cup \mathcal{A}) = P(H_k \cup \mathcal{A}|pa(H_k \cup \mathcal{A})); \text{Training: } H_k, \text{Pre-trained:} \mathcal{A} \tag{2}$$

Finally, we call a subroutine Algorithm 6: **TrainModule()**. This function employs conditional Wasserstein GAN with penalized gradients (Gulrajani et al., 2017) to train the generators in $\mathbb{G}_{H_k}$ on dataset $\mathbf{D}$ such that the joint distributions implied by the set $\mathcal{A}$ matches. This sub-routine uses all mechanisms in $H_k \cup \mathcal{A}$ to produce samples but only updates the mechanisms in $\mathbb{G}_{H_k}$ corresponding to the current h-node and returns those after convergence. In Figure 5, this sub-routine will match $P(z_1, z_2, z_3|x_1) = Q(z_1, z_2, z_3|\text{do}(x_1))$ at step (2/3) and $P(x_1, x_2, z_1, z_3) = Q(x_1, x_2, z_1, z_3)$ at step (3/3). After calling **TrainModule()** for each of the h-nodes according to the partial order of $\mathcal{H}$-graph, WhatIfGAN will find a DCM equivalent to the true SCM that matches $P(\mathcal{V})$ distribution. Furthermore, it will sample from $\mathcal{L}_2$ and $\mathcal{L}_3$ distributions identifiable from $P(\mathcal{V})$. These contributions are presented formally in Theorem 3.4. We provide the formal proof in Appendix C.3. ***WhatIfGAN sampling.*** For $\mathcal{L}_2$ sampling, we set the intervened variables to fixed values instead of using their neural network and push forward those values to generate the rest of the variables as usual. For $\mathcal{L}_3$ sampling of $P_x(Y|x')$, we follow i) Abduction: With rejection sampling, we record the posterior exogenous n_X, confounding U and the Gumbel noises that yield $X = x'$ in the DCM. ii) Action: We perform do(x) intervention in the DCM. iii) Prediction: We push forward pre-recorded exogenous noise as input to all neural networks.

C. Theoretical Analysis

C.1. Identifiable Distributions from WhatIfGAN Modular Training

In this section, we show that WhatIfGAN modular training will match the observational joint distribution $P(\mathcal{V})$. We start with some definitions that would be required during our proofs.

Definition C.1 (Intervention Set, $\mathcal{I}$)**.** Intervention Set, $\mathcal{I}$ represents the set of all available interventional variables such that after performing intervention $I \in \mathcal{I}$ on G, we observe $G_{\overline{I}}$. $\mathcal{I}$ includes $I = \emptyset$, which refers to "no intervention" and implies the original graph G and the observational data $P(\mathcal{V})$. All proofs here are considered for $I = \emptyset$.

Definition C.2 (Sub-graph, $(G)_V$)**.** Let G_V be a sub-graph of G containing nodes in V and all arrows between such nodes. $(G)_V$ refers to the sub-graph of G containing nodes in V only.

Proposition C.3. *Let $V \in \mathcal{V}$ be some arbitrary variable sets. The set of c-components formed from a sub-graph $(G)_V$ is not affected by additional interventions on their parents from outside of the sub-graph. Formally, $(G_{\overline{Pa(V)}})_V$ and $(G)_V$ has the same set of c-components.*

Proof. Let $C((G)_V)$ be the c-components which consists of nodes of V in graph $(G)_V$. In sub-graph $(G_{\overline{Pa(V)}})_V$, no extra intervention is being done on any node in V rather only on $Pa(V)$ where V and $Pa(V)$ are two disjoint sets. Therefore, the c-components can be produced from this sub-graph will be same as for G. i.e., $C(G_{V\overline{Pa(V)}}) = C(G)$. □

Lemma C.4. *Let V' be a set called focus-set. V' be arbitrary subsets of observable variables $\mathcal{V}$ and $\{C_i\}_i$ be the set of c-components in G. Let $Pa(V')$ be a set called action-set. and S be a set called remain-set, defined as $S := \mathcal{V} \backslash \{V' \cup Pa(V')\}$,*

$S(i)$ as $S(i) = S \cap C_i$ i.e., some part of the remain-set that are located in c-component C_i. Thus, $S = \bigcup_i S(i)$. We also define active c-components C_i^+ as $C_i^+ := C_i \setminus \{S(i) \cup Pa(V')\}$ i.e., the variables in focus-set that are located in c-component C_i. Given these sets, Tian's factorization can be applied to a sub-graph under proper intervention. Formally, we can factorize as below:

$$P_{Pa(V')}(V') = \prod_i P_{Pa(C_i^+)}(C_i^+)$$

Proof. $(G_{\overline{Pa(V')}})_{V'}$ and $(G)_{V'}$ have the same c-components according to Proposition C.3. According to Tian's factorization for causal effect identification (Tian & Pearl, 2002), we know that

$$\begin{aligned} P_{Pa(V')}(\mathcal{V}) &= \prod_i P_{Pa(C_i) \cup Pa(V')}(C_i) \\ &\text{[let } \eta = Pa(V')\text{, i.e., action-set]} \\ \implies P_\eta(\eta) \times P_\eta(\mathcal{V} \setminus \eta | \eta) &= \prod_i P_{Pa(C_i) \cup \eta}(C_i) \\ \implies P_\eta(\mathcal{V} \setminus \{Pa(V')\}) &= \prod_i P_{Pa(C_i) \cup \eta}(C_i) \end{aligned} \tag{3}$$

We ignore conditioning on action-set $\eta = Pa(V')$ since we are intervening on it. Now, we have a joint distribution of focus-set and remain-set with action-set as an intervention.

$$\begin{aligned} &\implies P_\eta(V' \cup S) = \prod_i P_{Pa(C_i) \cup \eta}(C_i) \\ &\text{[Here, } S := \mathcal{V} \setminus \{V' \cup Pa(V')\} \implies \mathcal{V} \setminus \{Pa(V')\} = V' \cup S\text{]} \\ &\implies \sum_S P_\eta(V' \cup S) = \sum_S \prod_i P_{Pa(C_i) \cup \eta}(C_i) \\ &\implies \sum_S P_\eta(V' \cup S) = \prod_i \sum_{S(i)} P_{Pa(C_i) \cup \eta}(C_i) \\ &\text{[Since, } S(i) = S \cap C_i \text{and } \forall (i,j), i \neq j, C_i \cap C_j = \emptyset \implies S_i \cap S_j = \emptyset\text{]} \end{aligned} \tag{4}$$

Here, $\forall_i, S(i)$ are disjoint partitions of the variable set S and contained in only c-component C_i, i.e, $S(i) = S \cap C_i$. Since $\forall_{i,j}, C_i \cap C_j = \emptyset$, this implies that $S_i \cap S_j = \emptyset$ would occur as well. Intuitively, remain-sets located in different c-components do not intersect. Therefore, each of the probability terms at R.H.S, $P_{Pa(C_i) \cup \eta}(C_i)$ is only a function of $S(i)$ instead of whole S. This gives us the opportunity to push the marginalization of $S(i)$ inside the product and marginalize the probability term. After marginalizing $S(i)$ from the joint, we define rest of the variables as active c-components C_i^+. The following figure helps to visualize all the sets. For our case, intervention $I_1, I_2 = \emptyset$

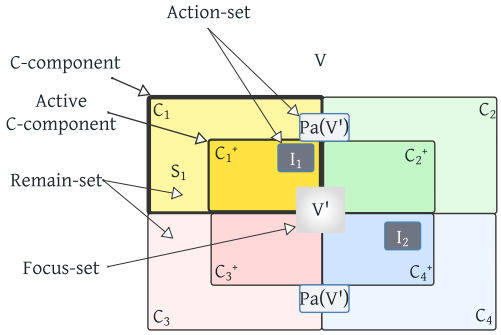


Figure 6. Visualization of focus-sets, action-sets, and remain-sets

We continue the derivation as follows:

$$\begin{aligned}
\implies P_{Pa(V')}(V') &= \prod_i P_{Pa(C_i)\cup Pa(V')}(C_i^+) \\
&\quad [Here, C_i^+ = C_i \setminus S(i), \text{i.e., active c-component: focus-set elements located in } C_i] \\
&= \prod_i P_{Pa(C_i^+)\cup\{Pa(C_i)\setminus Pa(C_i^+)\}\cup Pa(V')}(C_i^+) \\
&= \prod_i P_{X\cup Z}(C_i^+) \quad [\text{Let, } X = Pa(C_i^+) \text{ and } Z = \{Pa(C_i)\cup Pa(V')\}\setminus X]
\end{aligned} \tag{5}$$

Here, we have variable set C_i^+ in the joint distribution. Now, if we intervene on the parents $Pa(C_i^+)$, rest of the intervention which is outside C_i^+ becomes ineffective. Therefore, we have $X = Pa(C_i^+)$, the intervention which shilds the rest of the interventions, $Z = \{Pa(C_i) \cup Pa(V')\} \setminus X$. Therefore, we can apply do-calculus rule 3 on Z and remove those interventions. Finally,

$$\implies P_{Pa(V')}(V') = \prod_i P_{Pa(C_i^+)}(C_i^+) \quad [\text{We apply Rule 3 since } C_i \perp\!\!\!\perp Z | X_{G_{\overline{X}}}] \tag{6}$$

□

Corollary C.5, suggests that Tian's factorization can be applied on the h-nodes of $\mathcal{H}$.

Corollary C.5. *Consider a causal graph G. Let $\{C_i\}_{i\in[t]}$ be the c-components of G. Let $H = (V_\mathcal{H}, E_\mathcal{H})$ be the h-graph constructed by Algorithm 1 where $V_\mathcal{H} = \{H_k\}_k$. Suppose H_k is some node in $\mathcal{H}$. We have that $H_k = \{C_i\}_{i\in T_k}$ for some $T_k \subseteq [t]$. With slight abuse of notation we use H_k interchangeably with the set of nodes that are in H_k. Then,*

$$P_{Pa(H_k)}(H_k) = \prod_{i\in[t]} P_{Pa(C_i)}(C_i) \tag{7}$$

Proof. Let, $V' = H_k$, $C_i^+ = C_i \setminus \emptyset = C_i$. Then, this corollary is a direct application of Lemma C.4. □

C.2. Matching Distributions with WhatIfGAN Modular Training

Definition C.6 (Training order: $\mathcal{T}$)**.** We define a training order, $\mathcal{T} = \{\sigma_0, \ldots, \sigma_m\}$ where $\sigma_i = \{H_k\}_k$. If $H_{k_1} \to H_{k_2}, H_{k_1} \in \sigma_i, H_{k_2} \in \sigma_j$ then $i < j$.

Definition C.7 (Notation for distributions)**.** $Q(.)$ is the observational distribution induced by the deep causal SCM. $P(.)$ is the true (observational/interventional) distribution.

Definition C.8 (Ancestor set $\mathcal{A}$ in G)**.** Let parents of a variable set $\mathbf{V}$ be $Pa(\mathbf{V}) = \bigcup_{V\in\mathbf{V}} Pa(V) \setminus \mathbf{V}$. Now, for some h-node $H_k \in \mathcal{H}$-graph, we define $\mathcal{A} \coloneqq$ the minimal subset of ancestors exists in the causal graph G such that the following holds,

$$p(H_n \cup \mathcal{A}|\text{do}(pa(H_n \cup \mathcal{A})) = p(H_n \cup \mathcal{A}|pa(H_n \cup \mathcal{A})) \tag{8}$$

For training any h-node in the training order $\mathcal{T} = \{\sigma_0, \ldots, \sigma_m\}$, i.e., $H_k \in \sigma_j$, $0 < j \leq m$, if only observational data is available, we search for an ancestor set $\mathcal{A}$ such that $\mathcal{A}$ satisfies modularity condition for H_k:

$$P(H_k \cup \mathcal{A}|\text{do}(pa(H_k \cup \mathcal{A})) = P(H_k \cup \mathcal{A}|pa(H_k \cup \mathcal{A})) \tag{9}$$

Then we can train the mechanisms in H_k to learn the $P(\mathcal{V})$ c-factors by matching the following alternative distribution from D dataset,

$$\begin{aligned}
P(H_k \cup \mathcal{A}|pa(H_k \cup \mathcal{A})) &= Q(H_k \cup \mathcal{A}|\text{do}(pa(H_k \cup \mathcal{A}))) \\
\implies P(H_k \cup \mathcal{A}|\text{do}(pa(H_k \cup \mathcal{A})) &= Q(H_k \cup \mathcal{A}|\text{do}(pa(H_k \cup \mathcal{A})))
\end{aligned} \tag{10}$$

C.2.1. Matching Observational distributions with Modular Training on $D \sim P(\mathcal{V})$

Now, we provide the theoretical proof of the correctness of WhatIfGAN Modular Training matching observational distribution by training on observational dataset D. Since, we have access to only observational data we remove the intervention-indicating superscript/subscript and address $\mathcal{H}$ as $\mathcal{H}$, ancestor set $\mathcal{A}$ as $\mathcal{A}$ and dataset D as D.

Proposition C.9. *Suppose Algorithm 2: **WhatIfGAN Modular Training** converges for each h-node in $\mathcal{H}$-graph constructed from $G = (\mathcal{V}, \mathcal{E})$. Suppose the observational distribution induced by the deep causal model is $Q(\mathcal{V})$ after training on data sets $D \sim P(\mathcal{V})$. Then,*

$$P(\mathcal{V}) = Q(\mathcal{V}) \tag{11}$$

Proof. According to Tian's factorization we can factorize the joint distributions into c-factors as follows:

$$P(\mathcal{V}) = P(\mathcal{H}) = \prod_{H_k \in \mathcal{H}} \prod_{C_i \in H_k} P_{pa(C_i)}(C_i) \tag{12}$$

We can divide the set of c-components $\mathcal{C} = \{C_1, \dots C_t\}$ into disjoint partitions or h-nodes as $H_k = \{C_i\}_{i \in T_k}$ for some $T_k \subseteq [t]$. Following Corollary C.5, we can combine the c-factors in each partitions and rewrite it as:

$$\prod_{H_k \in \mathcal{H}} \prod_{C_i \in H_k} P_{pa(C_i)}(C_i) = P_{pa(H_0)}(H_0) \times P_{pa(H_1)}(H_1) \times \dots \times P_{pa(H_n)}(H_n) \tag{13}$$

Now, we prove that we match each of these terms according to the training order $\mathcal{T}$.

For any root h-nodes $H_k \in \sigma_0$:
Due to the construction of $\mathcal{H}$ graphs in Algorithm 1, the following is true for any root nodes, $H_k \in \sigma_0$.

$$P(H_k | Pa(H_k)) = P_{Pa(H_k)}(H_k) \tag{14}$$

WhatIfGAN training convergence for the DCM in $H_k \in \sigma_0$. (Algorithm 2 ensures that the following matches:

$$\begin{aligned} & P(H_k | Pa(H_k)) = Q_{Pa(H_k)}(H_k) \\ \implies & P_{Pa(H_k)}(H_k) = Q_{Pa(H_k)}(H_k) \end{aligned} \tag{15}$$

Since, Equation 14 is true, observational data is sufficient for training the mechanisms in $H_k \in \sigma_0$. Thus, we do not need to train on interventional data.

For the h-node $H_k \in \sigma_1$:
Now we show that we can train mechanisms in H_k by matching $P(\mathcal{V})$ c-factors with $D \sim P(\mathcal{V})$ data set. Let us assume, $\exists \mathcal{A} \subseteq \sigma_0$ such that $\mathcal{A} = An(H_k)$,i.e., ancestors set of H_k in the $\mathcal{H}$-graph that we have already trained with available D dataset. To apply Lemma C.4 in causal graph G, consider $V' = H_k \cup \mathcal{A}$ as the focus-set, $Pa(V')$ as the action-set. Thus, active c-components: $C_j^+ := C_j \cap V'$

Then we get the following:

$$\begin{aligned} P(H_k \cup \mathcal{A} | \text{do}(Pa(H_k \cup \mathcal{A}))) &= \prod_{C_i \in H_k} P_{pa(C_i)}(C_i) \times \prod_{H_S \in \{\mathcal{A}\}} \prod_{C_j^+ \subseteq H_S} P_{Pa(C_j^+)}(C_j^+) \\ & \qquad \text{[Here, 1st term is the factorization of the current h-node} \\ & \qquad \text{and 2nd term is the factorization of the ancestors set.]} \\ \implies P(H_k \cup \mathcal{A} | \text{do}(Pa(H_k \cup \mathcal{A}))) &= P_{pa(H_k)}(H_k) * \prod_{H_S \in \mathcal{A}} \prod_{C_j^+ \subseteq H_S} P_{Pa(C_j^+)}(C_j^+) \end{aligned} \tag{16}$$

Here according to Corollary C.5, we combine the c-factors $P_{pa(C_i)}(C_i)$ for c-components in H_k to form $P_{pa(H_k)}(H_k)$. We

continue the derivation as follows:

$$\begin{aligned} \implies P_{Pa(H_k)}(H_k) &= \frac{P(H_k \cup \mathcal{A}|\text{do}(Pa(H_k \cup \mathcal{A})))}{\prod_{H_S \in \mathcal{A}} \prod_{C_j^+ \subseteq H_S} P_{Pa(C_j^+)}(C_j^+)} \\ \implies P_{Pa(H_k)}(H_k) &= \frac{Q(H_k \cup \mathcal{A}|\text{do}(Pa(H_k \cup \mathcal{A})))}{\prod_{H_S \in \mathcal{A}} \prod_{C_j^+ \subseteq H_S} Q_{Pa(C_j^+)}(C_j^+)} \end{aligned} \tag{17}$$

Here the R.H.S numerator follows from previous line according to Equation 10. For the denominator at R.H.S, $\forall H_S \in \mathcal{A}$, we have already matched $P(H_S \cup \mathcal{A}|\text{do}(pa(H_S \cup \mathcal{A})))$, during training of $\mathcal{A} = An(H_k)$ h-nodes. According to Lemma C.4, matching these distribution is sufficient to match the distribution at R.H.S denominator. Therefore, our DCM will produce the same distribution as well. This implies that from Equation 17 we get,

$$P_{pa(H_k)}(H_k) = Q_{pa(H_k)}(H_k) \tag{18}$$

Similarly, we train each h-node following the training order $\mathcal{T}$ and match the distribution in Equation 13. This finally shows that,

$$P(V) = \prod_{j \leq n} P_{pa(H_j)}(H_j) = \prod_{j \leq n} Q_{pa(H_j)}(H_j) = Q(V) \tag{19}$$

□

C.3. Identifiability of Algorithm 2:WhatIfGAN Modular Training

Theorem C.10. *Suppose, modular training on Dataset D converges for each h-node in the $\mathcal{H}$-graph and DCM induces $Q(\mathcal{V})$. Then, for any interventional and counterfactual causal query $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V})$ identifiable from D, we have $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V}) = \mathcal{K}_{\mathcal{M}2}(\mathcal{V})$.*

Proof. Let $\mathcal{M}_1 = (G = (\mathcal{V}, \mathcal{E}), \mathcal{N}, \mathcal{U}, \mathcal{F}, P(.))$ be the true SCM and $\mathcal{M}_2 = (G, \mathcal{N}', \mathcal{U}', \mathcal{F}', Q(.))$ be the deep causal generative model represented by WhatIfGAN. For any $H_k \in \mathcal{H}$, we observe the joint distribution $P(H_k \cup \mathcal{A} \cup Pa(H_k \cup \mathcal{A}))$ in the input D datasets. Thus we can train all the mechanisms in the current h-node H_k by matching the following distribution from the partially observable datasets:

$$P(H_k \cup \mathcal{A}|pa(H_k \cup \mathcal{A})) = Q(H_k \cup \mathcal{A}|\text{do}(pa(H_k \cup \mathcal{A}))) \tag{20}$$

Now, as we are following a valid partial order of the $\mathcal{H}$-graph to train the h-nodes, we train the mechanisms of each h-node to match the input distribution only once and do not update it again anytime during the training of rest of the network. As we move to the next h-node of the partial order for training, we can keep the weights of the Ancestor h-nodes fixed and only train the current one and can successfully match the joint distribution in Equation 20. In the same manner, we would be able to match the distributions for each h-node and reach convergence for each of them. WhatIfGAN Training convergence implies that $Q(\mathcal{V}) = P(\mathcal{V})$ i.e., for all input dataset distributions. Therefore, according to Theorem A.4, WhatIfGAN is capable of producing samples from correct interventional or counterfactual distributions that are identifiable from the input distributions. □

Theorem C.11. *Suppose, modular training on Dataset D converges for each h-node in the $\mathcal{H}$-graph and DCM induces $Q(\mathcal{V})$. Then, we have $i) P(\mathcal{V}) = Q(\mathcal{V})$, and $ii)$ for any interventional and counterfactual causal query $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V})$ identifiable from D, we have $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V}) = \mathcal{K}_{\mathcal{M}2}(\mathcal{V})$.*

Proof. Theorem 3.4 is restated here. The first part of the theorem is proved in Proposition C.9. The second part can be proved with Theorem C.10. □

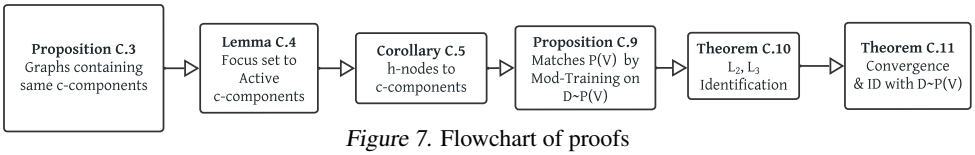


Figure 7. Flowchart of proofs

D. Experimental Evaluation

D.1. Training Details and Compute

We performed our experiments on a machine with RTX-3090 GPU. The experiments took 1-4 hours to complete. We ran the experiment for 300 epochs. We repeated each experiment multiple times to observe the consistent behavior. Our dataset contained $20K$ samples and batch_size 200 using the ADAM optimizer. After a few epochs, we generated 20k fake samples. Next, we calculated the required distributions from the real dataset and 20k fake samples. We calculated TVD and KL distance between these two distributions. For Wassertein GAN with gradient penalty, we used LAMBDA_GP=10. We had learning_rate= $5 * 1e-4$. We used Gumbel-softmax with a temperature starting from 1 and decreasing till it is 0.1. We used different architectures for different experiments since each experiment dealt with different data types: low-dimensional and image. Details are provided in the code. For low dimensional variables, we used $input\_dim \times 256 \times 256 \times output\_dim$. with BatchNorm and ReLU between each layer. Please check our code for architectures of other neural networks such as encoders and image generators.

D.2. Image Mediator Experiment

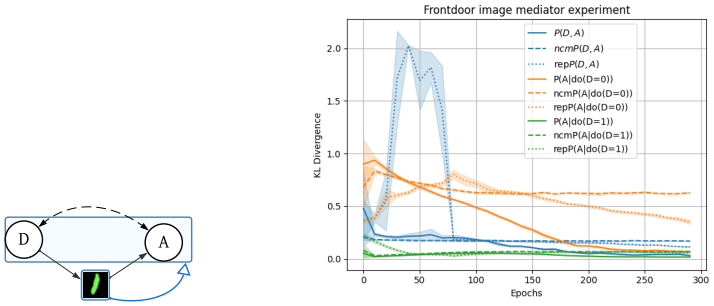


Figure 8. Left: Frontdoor causal graph w/ image mediator. Right: Modular Training on frontdoor causal graph with training order: $\{I\} \rightarrow \{D, A\}$ Training converges matching $P(D, A)$ and $P(A|do(D))$.

In this section, we provide additional information about the experiment described in Section 4. We have domain $D = [0, 1]$, Image size=$3 \times 32 \times 32$ and $C = [0, 1, 2]$. Let U_0, e_1, e_2, e_3 are randomly generate exogenous noise. $D = U_0 + e_1, Image = f_2(D, e_2), C = f_3(Image, e_3, U_0)$. f_2 is a function which takes D and e_2 as input and produces different colored images showing D digit in it. f_3 is a classifier with random weights that takes U_0 and $Image$ as input and produces C such a way that $|P(C|do(D = 0)) - P(C|D = 0)|, |P(C|do(D = 1)) - P(C|do(D = 0))|and|P(C|D = 1) - P(C|D = 0)|$ is high. We calculate ground truth of P(Cldo(D)) with backdoor criterion.

$$P(C|do(D)) = \sum_{U_0} P(C|D, U_0)P(D|U_0)$$

. WhatIfGAN samples from $P(C|do(D))$ after training. The query is identifiable with frontdoor criterion when U_0 is unobserved. Image is a mediator here.

$$P(C|do(D)) = \sum_{Image} P(Image|D) \sum_{D'} P(C|D', Image)P(D')$$

This inference is not possible with identification algorithm. WhatIfGAN can achieve that by producing Image samples instead of learning its distribution. Training We use a lower-dimensional representation RI of Image variable and fit $P(D, RI, C)$ instead of $P(D, Image, C)$. Samples from $P(Image|do(D = 1))$

E. Algorithms & Pseudo-codes

Algorithm 3 isIdentifiable($G, \mathcal{I}, query$)

Input: Causal Graph $G = (\mathcal{V}, \mathcal{E})$, Interventions = I, Causal query distribution= $query$
if type($query$)=Counterfactual **then**
  Return Run_IDC($G, query, I$)
else if type($query$)=Interventional **then**
  Return Run_ID($G, query$) or hasSurrogates($G, query, I$)
end if

Algorithm 4 RunGAN($G, \mathbb{G}, V_{\mathbf{K}}, I, N$)

Input: Causal Graph $G = (\mathcal{V}, \mathcal{E})$, DCM $\mathbb{G}$, target variable set $V_{\mathbf{K}}$, Intervention I, Pre-defined noise N.
for $V_i, V_j \in V_{\mathbf{K}}$ such that $i < j$ **do**
  if V_i, V_j has latent confounder **then**
    $z \sim p(z)$
    $conf[V_i] \leftarrow Append(conf[V_i], z)$
    $conf[V_j] \leftarrow Append(conf[V_j], z)$ // Assigning same confounding noise [fix for multiple confounders]
  end if
end for
for $V_i \in V_{\mathbf{K}}$ in causal graph,G topological order **do**
  if $V_i \in I.keys()$ **then**
    $v_i = I[V_i]$ // Assigning intervened value
  else
    $par = get\_parents(V_i, G)$
    if $V_i \in N.keys()$ **then**
      $exos, conf, gumbel = N[V_i]$
    else
      $exos \sim p(z)$
      $conf = conf[V_i]$
      $gumbel = \emptyset$. //New Gumbel noise will be assigned during forward pass
    end if
    $v_i = \mathbb{G}_{\theta_i}(exos, conf, gumbel, \hat{\mathbf{v}}_{par})$
  end if
  $\hat{\mathbf{v}} \leftarrow Append(\hat{\mathbf{v}}, v_i)$
end for
Return Samples **v** or Fail

Algorithm 5 Evaulate_GAN($G, \mathbb{G}, \mathcal{I}, query$)

Input: Causal Graph $G = (\mathcal{V}, \mathcal{E})$, DCM= $\mathbb{G}$, Available Interventions = $\mathcal{I}$, Causal query distribution=$query$
if isIdentifiable($G, \mathcal{I}, query$) = False **then**
  Return: Fail
end if
if type(query)= observation **then**
  $Y = Extract(query)$
  $samples \leftarrow$ RunGAN($G, \mathbb{G}, [Y], \emptyset, \emptyset$)
else if type(query)= Intervention **then**
  $Y, (X, x) :=$ Extract($query$)
  $samples \leftarrow$ RunGAN($G, \mathbb{G}, [Y], \{X : x\}, \emptyset$)
else if type(query)= Counterfactual **then**
  $Y, (X, x), (X, x') :=$ Extract($query$)
  $exos, conf, gumbel \leftarrow$ RejectionSampling($\{X : x'\}$)
  $N \leftarrow [exos, conf, gumbel]$
  $samples \leftarrow$ RunGAN($G, [Y], \{X : x\}, N$)
end if
Return $samples$

Algorithm 6 TrainModule($\mathbb{G}, G, H_*, \mathcal{A}, \mathbf{D}$)

Input: DCM $\mathbb{G}$, Graph $G(\mathcal{V}, \mathcal{E})$, h-node H_*, Ancestor set $\mathcal{A}$, Data $\mathbf{D}$, Params $\theta_H, \lambda = 10$
while θ_{H_*} has not converged **do**
  for each $(\mathcal{A}_i, X_i, D_i) \in (\mathcal{A}, \mathbf{D})$ **do**
    $V_r = H_* \cup \mathcal{A}_i \cup Pa(H_* \cup \mathcal{A}_i) \cup X_i$
    Initialize critic $\mathbb{D}_{w_i}$
    for $t = 1, \ldots, m$ {m samples} **do**
      Sample real data $\mathbf{v}_{\mathbf{x}}^{\mathbf{r}} \sim D_i$
      $\mathbf{x}^r \leftarrow$ get_intv_values(X_i, D_i)
      $\mathbf{v}_x^f =$ RunGAN($\mathbb{G}, \mathbf{x}^r, V_r, \theta_{H_*}$)
      $\hat{\mathbf{v}}_x = \epsilon \mathbf{v}_x^r + (1-\epsilon)\mathbf{v}_x^f$
      $L_i^{(t)} = \mathbb{D}_{w_i}(\mathbf{v}_x^f) - \mathbb{D}_{w_i}(\mathbf{v}_x^r) * \lambda(\|\nabla_{\hat{\mathbf{v}}_x} \mathbb{D}_{w_i}(\hat{\mathbf{v}}_x)\|_2 - 1)^2$
    end for
    $w_i = Adam(\nabla_{w_i} \frac{1}{m} \sum_{t=1}^{m} L_i^{(t)}, w_i)$
    $G_{loss} = G_{loss} + \frac{1}{m} \sum_{j=1}^{m} -\mathbb{D}_{w_i}(\mathbf{v}_x^f)$
  end for
  for $\theta \in \theta_{H_*}$ {All hnode mechanisms} **do**
    $\theta = Adam(\nabla_\theta G_{loss}, \theta)$
  end for
end while
Return: $\theta_1, \ldots \theta_n$

Algorithm 7 IsRule2($Y, X, I = \emptyset$ (by default))

Input: Variable sets Y and X, Intervention I.
Return:
if $P(Y \cup X|\text{do}(Pa(Y \cup X)), \text{do}(I)) = P(Y \cup X|Pa(Y \cup X), \text{do}(I)) = True$ **then**
  Return:1
else
  Return:0
end if