Modular Learning of Deep Causal Generative Models for High-dimensional Causal Inference

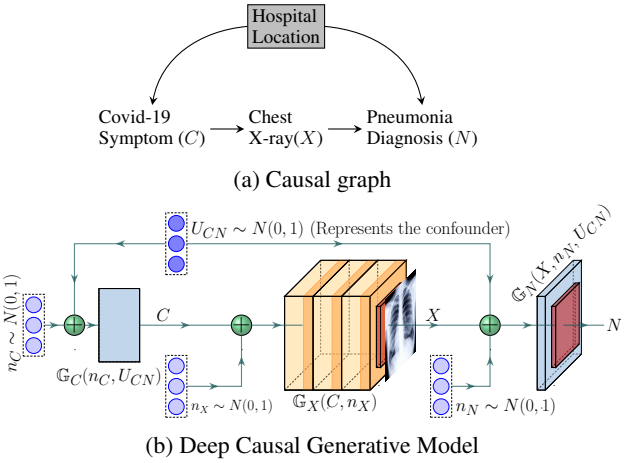
Md Musfiqur Rahman [1] **Murat Kocaoglu** [1]

Abstract

Sound and complete algorithms have been proposed to compute identifiable causal queries using the causal structure and data. However, most of these algorithms assume accurate estimation of the data distribution, which is impractical for high-dimensional variables such as images. On the other hand, modern deep generative architectures can be trained to sample from high-dimensional distributions. However, training these networks are typically very costly. Thus, it is desirable to leverage pre-trained models to answer causal queries using such high-dimensional data. To address this, we propose modular training of deep causal generative models that not only makes learning more efficient, but also allows us to utilize large, pre-trained conditional generative models. To the best of our knowledge, our algorithm, Modular-DCM is the first algorithm that, given the causal structure, uses adversarial training to learn the network weights, and can make use of pre-trained models to provably sample from any identifiable causal query in the presence of latent confounders. With extensive experiments on the Colored-MNIST dataset, we demonstrate that our algorithm outperforms the baselines. We also show our algorithm's convergence on the COVIDx dataset and its utility with a causal invariant prediction problem on CelebA-HQ.

1. Introduction

Evaluating the causal effect of an intervention on a system of interest is one of the fundamental questions that arise across disciplines. Pearl's structural causal models (SCMs) provide a principled approach to answering such queries from data. Using SCMs, today we have a clear understanding of which causal queries can be answered from data, and which cannot without further assumptions (Pearl, 1995; Shpitser & Pearl, 2008; Huang & Valtorta, 2012; Bareinboim & Pearl, 2012b). These *identification algorithms* find a closed-form expression for an interventional distribution using only observational data, by making use of the causal structure via do-calculus rules (Pearl, 1995). Today, most of our datasets contain high-dimensional variables, such as images. Modern deep learning architectures can handle high-dimensional data and solve *non-causal* machine learning problems such as classification, detection or generation. The existing causal inference algorithms can answer any identifiable causal question, but they cannot handle high-dimensional variables as they require access to the joint distribution, which is not practical with image data.



(a) Causal graph

(b) Deep Causal Generative Model

Figure 1. Causal graph for the XrayImg example (top) and its deep causal generative model (bottom). For each variable, an NN ($\mathbb{G}_C, \mathbb{G}_X, \mathbb{G}_N$) is trained to mimic the true mechanism.

As an example, consider a healthcare dataset of Covid symptoms (C), Pneumonia diagnosis (N) and chest X-rays (X) of patients from hospitals in two cities with different socio-economic status, which we do not observe to ensure patient privacy. Then, hospital location acts as a latent confounder for both C and N since it might have effect on how likely pa-

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

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

tients are getting sick on average and if they have access to better trained doctors. In this scenario, the data-generating process can be summarized by the causal graph in Figure 1. We would like to understand how likely an average person across the two cities is to be diagnosed with pneumonia if they have Covid symptoms, i.e., the causal effect of Covid symptoms on pneumonia diagnosis, $P_C(N)$. The causal effect can be computed from the observational distribution as $\int_{x,c'} p(x|c)p(n|x,c')p(c')$ through the *front-door adjustment* (Pearl, 2009). However, it is not possible to reliably estimate $P(x|c)$ or marginalize over all possible $\mathrm{XrayImage}$ due to its high dimensionality. To the best of our knowledge, no existing algorithm can address this problem.

Although the use of such structured deep generative models has been explored recently, existing solutions either assume no unobserved confounders (Kocaoglu et al., 2018; Pawlowski et al., 2020), consider specific graphs (Louizos et al., 2017; Zhang et al., 2021) or are effective for low-dimensional variables (Xia et al., 2021; 2023) due to the challenge of learning high-dimensional joint distribution. Furthermore, these methods do not have the flexibility to use pre-trained models without affecting their weights. This is useful since state-of-the-art deep models, such as large image generators, can only be successfully trained by a few industrial research labs with expensive resources.

In this paper, we propose a modular, sampling-based solution to address the high-dimensionality challenge the existing algorithms face while learning deep causal generative models. Our solution uses deep learning architectures that mimic the causal structure of the system such as in Figure 1b. We offer efficient and flexible training facilitated by our key contribution: the ability to identify which parts of the deep causal generative models can be trained separately (such as $\{G_X\}$ in Figure 1b), and which parts should be trained together ($\{G_S, G_D\}$). We show that after this modularization, there is a *correct* training order for each sub-network (ex:$\{G_X\} \rightarrow \{G_S, G_D\}$). Our algorithm follows such an order to train the sub-networks while freezing weights of already trained ones in the previous steps. After training, our method can be used to obtain samples from the interventional distribution (ex: $P_C(N)$), which is implicitly modeled. Thus the modularity in our method enables the flexibility to plug in pre-trained generative networks, and paves the way to utilize large pre-trained models for causal inference. The following are our main contributions.

- We propose an adversarial learning algorithm for training deep causal generative models with latent confounders for high-dimensional variables. We show that, after convergence, our model can produce high-dimensional samples according to interventional queries that are identifiable from the data distributions.
- To the best of our knowledge, ours is the first algorithm that can modularize the training process in the presence of latent confounders while preserving their theoretical guarantees, thereby enabling the use of large pre-trained models for causal effect estimation.
- With extensive experiments on Colored-MNIST, we demonstrate that Modular-DCM converges better compared to the closest baselines and can correctly generate interventional samples. We also show our convergence on COVIDx CXR-3 and solve an invariant prediciton problem on CelebA-HQ.

2. Related Works

In recent years, a variety of neural causal methods have been developed in the literature to answer interventional queries (Gao et al., 2022; Jerzak et al., 2022; Dash et al., 2022; Qin et al., 2021; Castro et al., 2020). Shalit et al. (2017); Louizos et al. (2017); Nemirovsky et al. (2020) offer to solve the causal inference problem using deep generative models. Yet, they do not offer theoretical guarantees of causal estimation in general, but for some special cases.

Researchers have recently focused on imposing causal structures within neural network architectures. Particularly, Kocaoglu et al. (2018) introduced a deep causal model that produces interventional image samples after training on observational data. Chao et al. (2023) offer similar contribution with diffusion-based (Song et al., 2020) approaches. Pawlowski et al. (2020); Ribeiro et al. (2023) apply normalizing flows and variational inference to predict exogenous noise for counterfactual inference. Dash et al. (2022) use an encoder and a generator to produce counterfactual images in order to train a fair classifier. A major limitation of these works is the causal sufficiency assumption, i.e., each variable is caused by *independent* unobserved variables. Semi-Markovian models (Tian et al., 2006) which allow unobserved confounders affect pairs is more practical.

For semi-Markovian models, Xia et al. (2021); Balazadeh Meresht et al. (2022); Xia et al. (2023) follow a similar approach as Kocaoglu et al. (2018) to arrange neural models as a causal graph. They propose a minimization-maximization method to identify and estimate causal effects. Xia et al. (2023) extend these to identify and estimate counterfactual queries.

Most of the existing methods described above can handle only discrete or low-dimensional (Bica et al., 2020; Yang et al., 2021) variables and it is not clear how to extend their results to continuous high-dimensional image data. Moreover, if these methods are given a pre-trained neural network model, they do not have the ability to incorporate them in their training. To the best of our knowledge, our approach is the first to address this problem in the presence of unobserved confounders, unlocking the potential of large

pre-trained models for causal inference.

3. Background

Definition 3.1 (Structural causal model (SCM) (Pearl, 2009))**.** An SCM $\mathcal{M}$ is a 5-tuple $\mathcal{M} = (\mathcal{V}, \mathcal{N}, \mathcal{U}, \mathcal{F}, P(.))$, where each observed variable $V_i \in \mathcal{V}$ is realized as an evaluation of the function $f_i \in \mathcal{F}$ which looks at a subset of the remaining observed variables $Pa_i \subset \mathcal{V}$, an unobserved exogenous noise variable $E_i \in \mathcal{N}$, and an unobserved confounding (latent) variable $U_i \in \mathcal{U}$. $P(.)$ is a product joint distribution over all unobserved variables $\mathcal{N} \cup \mathcal{U}$.

Each SCM induces a directed graph called the *causal graph*, or acyclic directed mixed graph (ADMG) with $\mathcal{V}$ as the vertex set. The directed edges are determined by which variables directly affect which other variable by appearing explicitly in that variable's function. Thus the causal graph is $G = (V, E)$ where $V_i \to V_j$ iff $V_i \in Pa_j$. The set Pa_j is called the parent set of V_j. We assume this directed graph is acyclic (DAG). Under the semi-Markovian assumption, each unobserved confounder can appear in the equation of exactly two observed variables. We represent the existence of an unobserved confounder between X, Y in the SCM by adding a bidirected edge $X \leftrightarrow Y$ to the causal graph. These graphs are no longer DAGs although still acyclic.

V_i is called an ancestor for V_j if there is a directed path from V_i to V_j. Then V_j is said to be a descendant of V_i. The set of ancestors of V_i in graph G is shown by $An_G(V_i)$. A do-intervention $do(v_i)$ replaces the functional equation of V_i with $V_i = v_i$ without affecting other equations. The distribution induced on the observed variables after such an intervention is called an interventional distribution, shown by $P_{v_i}(\mathcal{V})$. $P_\emptyset(\mathcal{V}) = P(\mathcal{V})$ is called the observational distribution. In this paper, we use $\mathcal{L}_1$ and $\mathcal{L}_2$ as notation for observational and interventional distributions, respectively.

Definition 3.2 (c-components)**.** A subset of nodes is called a c-component if it is a maximal set of nodes in G that are connected by bi-directed paths.

4. Deep Causal Generative Model with Latents

Suppose the ground truth data-generating SCM is made up of functions $X_i = f_i(Pa_i, E_i)$. If we have these equations, we can simulate an intervention on, say $X_5 = 1$, by evaluating the remaining equations. However, we can never hope to learn the true functions and unobserved noise terms from data. The fundamental observation of Pearl is that even then there are some causal queries that can be uniquely identified as some *deterministic function* of the causal graph and the joint distribution between observed variables, e.g., $p(n|do(c)) = \xi(G, p(c, x, n))$ in Figure 1a for some deterministic ξ. This means that, if we can, somehow, train a causal model made up of neural networks that fits the $data \sim p(\mathbf{v})$, and has the same causal graph, then it has to induce the same interventional distribution $p(d|do(s))$ as the ground truth SCM, *irrespective of what functions the neural network uses*. This is a very strong idea that allows mimicing the causal structure, and opens up the possibility of using deep learning algorithms for performing causal inference through sampling even with high-dimensional variables. This is the basic idea behind (Kocaoglu et al., 2018) without latents. Motivated by their work, we define a *deep causal generative model* for semi-Markovian model and show identifiability [1] results. Now, we formalize the above simple observations.

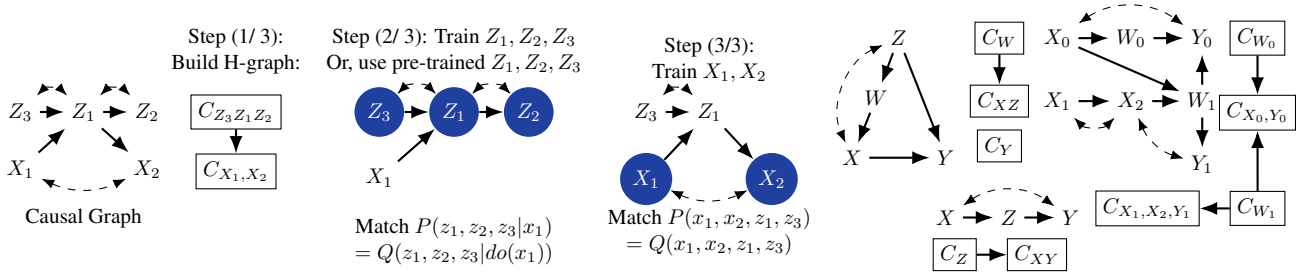
Definition 4.1 (DCM)**.** A neural net architecture $\mathbb{G}$ is called a deep causal generative model (DCM) for an ADMG $G = (\mathcal{V}, \mathcal{E})$ if it is composed of a collection of neural nets, one $\mathbb{G}_i$ for each $V_i \in \mathcal{V}$ such that i) *each $\mathbb{G}_i$ accepts a sufficiently high-dimensional noise vector N_i,* ii) *the output of $\mathbb{G}_j$ is input to $\mathbb{G}_i$ iff $V_j \in Pa_G(V_i)$,* iii) *$N_i = N_j$ iff $V_i \leftrightarrow V_j$.*

We define Q as the distribution induced by the DCM. Noise vectors N_i replace both the exogenous noises and the unobserved confounders in the true SCM. They are of sufficiently high dimension to induce the observed distribution. We say that a DCM is *representative enough for an SCM* if the neural networks have sufficiently many parameters to induce the observed distribution induced by the SCM. For the neural architectures of variables in the same c-component, we can consider conditional GANs (Mirza & Osindero, 2014), as they are effective in matching the joint distribution by feeding the same prior noise $N_i = N_j$ (as confounders) into multiple generators. For variables that are not confounded ($N_i \neq N_j$), we can use conditional models such as diffusion models (Ho & Salimans, 2022). With Defintion4.1, we have the following, similar to (Xia et al., 2021):

Theorem 4.2. *Consider any SCM $\mathcal{M} = (G, \mathcal{N}, \mathcal{U}, \mathcal{F}, P(.))$. A DCM $\mathbb{G}$ for G entails the same identifiable interventional distributions as the SCM $\mathcal{M}$ if it entails the same observational distribution.*

Thus, even with high-dimensional variables in the true SCM, given a causal graph, in principle, any identifiable interventional query can be sampled from, with a DCM that fits the observational distribution. However, to learn the DCM, (Kocaoglu et al., 2018; Xia et al., 2021) suggest training all neural nets $\mathbb{G}$ in the DCM together. Such an approach to match the joint distribution containing all low and high-dimensional variables is empirically challenging in terms of convergence. Any modularization not only is expected to help train more efficiently for better solution quality, but also allow the flexibility to use pre-trained image generative models. Now, we focus on uncovering how to achieve such modularization and how it contributes along the two aspects.

[1] Identifiability here refers to our ability to uniquely sample from an interventional distribution. See Definition C.1 for details.



(a) Step 1: Since $P(x_1, x_2|do(z_1)) \neq P(x_1, x_2|z_1)$, add edge from c-component $H_1 : [Z_1, Z_2, Z_3]$ to c-component $H_2 : [X_1, X_2]$. Step 2: train only networks in H_1 to match $P(z_1, z_2, z_3|x_1) = Q(z_1, z_2, z_3|do(x_1))$. Step 3: train only H_2 while using pre-trained Z_1, Z_3 to match $P(x_1, x_2, z_1, z_3) = Q(x_1, x_2, z_1, z_3)$.

(b) The rectangular box represents each h-node and which networks we have to train together. The partial order of the h-graph represents in which order we should train the mechanisms

Figure 2. (Left:) Modular training in 3 steps. (Right:) Causal graphs and their h-graphs showing modularization of the training process.

4.1. Modular-DCM Intuitive Explanation

Consider the graph G in Figure 2a. Suppose, we have an observational dataset $D \sim P(\mathcal{V})$. Based on Theorem 4.2, we can to sample from different $\mathcal{L}_2$ distributions such as $P(x_2|do(x_1))$ and $P(z_2|do(x_1))$ by training a DCM $\mathbb{G}$ that is consistent with G and fits the observational data $P(\mathcal{V})$. The DCM will contain one feed-forward neural net per observed variable, i.e., $\mathbb{G} = \{\mathbb{G}_{Z_1}, \mathbb{G}_{Z_2}, \mathbb{G}_{Z_3}, \mathbb{G}_{X_1}, \mathbb{G}_{X_2}\}$. If we follow the naive way and jointly train all networks in $\mathbb{G}$ together, we have to match $P(x_1, x_2, z_1, z_2, z_3)$ containing all low and high dimensional variables in a single training phase. Matching this joint distribution by training all models at the same time could be difficult, since we are attempting to minimize a very complicated loss function. Thus, the question we are interested in is, which neural nets can be trained separately, and which need to be trained together *to be able to fit the joint distribution*.

Suppose we first train the causal generative model $\mathbb{G}_{Z_1}$, i.e., learn a mapping that can sample from $P(z_1|z_3, x_1)$. Even if we provide the unobserved confounder N_1 (Definition 4.1), which also affects Z_2 and Z_3, the neural network might learn a mapping that later makes it impossible to induce the correct dependence between Z_1, Z_2, or Z_1, Z_3 no matter how $\mathbb{G}_{Z_2}$ or $\mathbb{G}_{Z_3}$ are trained later. This is because fitting the conditional $P(Z_1|Z_3, X_1)$ does not provide any incentive for the model $\mathbb{G}_{Z_1}$ to induce the correct confounding dependency (through the latent variables) with Z_2 and Z_3. If the model ignores the confounding dependence, it cannot induce the dependence between Z_3 and Z_2 conditioned on Z_1 ($Z_3 \not\perp\!\!\!\perp Z_2|Z_1$). This 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; similarly $[\mathbb{G}_{X_1}, \mathbb{G}_{X_2}]$ together.

To match the joint $P(\mathcal{V})$ 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 C_j intervened on their parents, i.e., $P_{pa(c_j)}(c_j)$.

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

Due to this factorization, fitting $P(\mathcal{V})$ is equivalent to fitting each of the c-factors. If we had access to the $\mathcal{L}_2$ distributions from do(z_1) and 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 do(z_1) and do(x_1) samples and train the models to match these $\mathcal{L}_2$ distributions. However, we only have access to the $P(\mathcal{V})$ dataset.

Note that it is very difficult to *condition* in feedforward models during training, which is the case in a DCM. To sample from $Q(x_2|z_1)$ it is not sufficient to feed z_1 to the network $\mathbb{G}_{X_2}$. In fact, observe that this is exactly the intervention operation, and would give us a sample from $Q(x_2|do(z_1))$. It is trivial to intervene on the inputs to a neural network, but highly non-trivial to condition since feedforward models cannot easily be used to correctly update the posterior via backdoor paths. Thus, we need to find some interventional distribution such that the DCM can generate samples for this distribution and can be trained by comparing them with some equivalent true observational samples.

Our key idea is to **leverage the do-calculus rule-2** (Pearl, 1995) to use observational samples and pretend that they are from these $\mathcal{L}_2$ distributions. This gives us a handle on how to modularize the training process of c-components. For example, in Figure 2a, c-factor $P(z_1, z_2, z_3|\text{do}(x_1)) = P(z_1, z_2, z_3|x_1)$ 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/alternative to the c-factor to learn $Q(z_1, z_2, z_3|\text{do}(x_1))$ with the DCM. However, $P(x_1, x_2|\text{do}(z_1)) \neq P(x_1, x_2|z_1)$. To overcome this issue, we seek to fit a joint distribution that implies this c-factor, i.e., we find a superset of X_1, X_2 on which rule-2 applies. We can include Z_1 into the joint distribution that needs to be matched together with X_1, X_2 and check if the parent set of $\{X_1, X_2, Z_1\}$ satisfy rule-2. We continue

including variables 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.

After identifying which sub-networks of the DCM can be trained separately, we need to decide *a valid order* in which they should be trained. For the same example, we can first train $[\mathbb{G}_{Z_1}, \mathbb{G}_{Z_2}, \mathbb{G}_{Z_3}]$ together to induce $Q(z_1, z_2, z_3|do(x_1)) = P(z_1, z_2, z_3|x_1) = P(z_1, z_2, z_3|do(x_1))$. This is shown in step (2/3) in Figure 2a: 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. Now, we train mechanisms of the next c-component $[\mathbb{G}_{X_1}, \mathbb{G}_{X_2}]$ in our training order (step 3/3). As discussed, we need to ensure $Q(x_1, x_2, z_1, z_3|do(\emptyset)) = P(x_1, x_2, z_1, z_3|do(\emptyset)) = P(x_1, x_2, z_1, z_3|\emptyset)$. Since mechanisms of Z_1, Z_3 were trained in the previous step, we can freeze the network weights of $[\mathbb{G}_{Z_1}, \mathbb{G}_{Z_3}]$. These are used to correctly sample from Z_1 given X_1, and feed this correctly sampled value into the network of X_2. In Appendix D.1, we show that the c-factors in Equation 1 will correctly match the true c-factors after fitting these two conditional probabilities in this order. Therefore, DCM matches the joint distribution $P(\mathcal{V})$ as well. On the other hand, if we first trained the networks $[\mathbb{G}_{X_1}, \mathbb{G}_{X_2}]$, it would not be possible to match the joint $P(x_1, x_2, z_1, z_3)$ as the mechanisms of Z_1, Z_3 are not yet trained. Thus, this order would not work.

4.2. Training Algorithm for Modular-DCM

In this section, we generalize the discussed ideas, into a modular algorithm that has mainly two phases: 1) arranging the c-components in a valid training order and 2) training (sets of) c-components to match their c-factors.

Arranging the c-components: Consider a c-component C_t. When should a c-component C_s be trained before C_t? Since we need rule 2 of do-calculus to hold on the parents of C_t for training, if C_s contains some parents of C_t that are located on the backdoor paths between any two variables in C_t, then C_s must be pre-trained before C_t. Conditioning and intervening on those parents of C_t is not the same, i.e., $P(C_t|do(pa(C_t) \cap C_s) \neq P(C_t|pa(C_t) \cap C_s)$. Thus we include $pa(C_t) \cap C_s$ in the joint distribution that we want to match for C_t, which requires those parents in C_s to be pre-trained. For the front-door graph in Figure 2b, we observe that $P(X, Y|\text{do}(Z)) \neq P(X, Y|Z)$. Thus, we train $\mathbb{G}_X$, $\mathbb{G}_Z$, $\mathbb{G}_Y$ in $[C_Z : \{\mathbb{G}_Z\}] \rightarrow [C_{XY} : \{\mathbb{G}_X, \mathbb{G}_Y\}]$ order.

To obtain a partial order among all c-components, we construct a directed graph structure called $\mathcal{H}$-graph that contains c-components as nodes. While adding edges, if any cycle is formed, we merge c-components on that cycle into a single h-node indicating that they will need to be trained jointly. Thus some h-nodes may contain more than one c-component. The final structure is a DAG and gives us a valid partial order $\mathcal{T}$ for modular training (Proposition D.14). Formally, an $\mathcal{H}$-graph is defined as:

Algorithm 1 Modular Training(G, $\mathbf{D}$)

1: **Input:** Causal Graph G, Dataset $\mathbf{D}$.
2: Initialize DCM $\mathbb{G}$
3: $\mathcal{H} \leftarrow$ Construct_Hgraph(G)
4: **for each** $H_k \in \mathcal{H}$ in partial order **do**
5: Initialize $\mathcal{A}_k \leftarrow \emptyset$
6: **while** IsRule2($H_k, \mathcal{A}_k$) $= 0$ **do**
7: $\mathcal{A}_k \leftarrow Pa_G(H_k, \mathcal{A}_k)$
8: $\mathbb{G}_{H_k} \leftarrow$TrainModule($\mathbb{G}_{H_k}, G, H_k, \mathcal{A}_k, \mathbf{D}$)
9: **Return:** $\mathbb{G}$

Definition 4.3 ($\mathcal{H}$-graph)**.** Given a causal graph G with c-components $\mathcal{C} = \{C_1, \dots C_n\}$, let $\{H_k\}_k$ be some partition of $\mathcal{C}$. The directed graph $(V_\mathcal{H}, E_\mathcal{H})$ where $V_\mathcal{H} = \{H_k\}_k$ and $H_s \rightarrow H_t \in E_\mathcal{H}$ iff $P(H_t|\text{do}(pa_G(H_t) \cap H_s)) \neq P(H_t|pa_G(H_t) \cap H_s)$, is called an $\mathcal{H}$-graph for G if acyclic.

We run Algorithm 7:Contruct_Hgraph() to build an $\mathcal{H}$-graph by checking the edge condition on line 5. In Figure 2 and Appendix E.1, we provide some examples of $\mathcal{H}$-graphs. Note that we only use the $\mathcal{H}$-graph to obtain a partial training order of h-nodes. For any h-node H_k, $An(H_k)$ and $Pa(H_k)$ below refer to ancestors and parents in the causal graph G, not in the $\mathcal{H}$-graph. ***Training c-components***: We follow $\mathcal{H}$-graph's topological order and train the c-components in an h-node H_k. If we can match $P_{pa(H_k)}(H_k)$, it will ensure that the DCM will learn their corresponding c-factors $P_{pa(C_j)}(C_j), \forall C_j \in H_k$ as well. As mentioned earlier, we can generate fake interventional samples from $Q_{pa(H_k)}(H_k)$ induced by the DCM, but they cannot be used to train $\mathbb{G}_{H_k}$ as we do not have access to real data samples from the interventional distribution $P_{pa(H_k)}(H_k)$. Thus, we train $\mathbb{G}_{H_k}$ to learn a larger joint distribution that can be obtained from the observational dataset as an alternative to its c-factors. We search for a set $\mathcal{A}_k$ that can be added to the joint with H_k such that $P_{Pa(H_k, \mathcal{A}_k)}(H_k, \mathcal{A}_k) = P(H_k, \mathcal{A}_k|Pa(H_k, \mathcal{A}_k))$, i.e., true interventional and conditional distribution are the same. This enables us to take conditional samples from the input dataset and use them as true interventional samples to match them with the DCM-generated fake interventional samples from $Q_{Pa(H_k, \mathcal{A}_k)}(H_k, \mathcal{A}_k)$ and train $\mathbb{G}_{H_k}$. The above condition is generalized as a **modularity condition**:

Definition 4.4. Let H_k be an h-node in the $\mathcal{H}$-graph. A set $\mathcal{A}_k \subseteq An_G(H_k) \setminus H_k$ satisfies the modularity condition if it is the smallest set with $P(H_k, \mathcal{A}_k|do(Pa(H_k, \mathcal{A}_k))) = P(H_k, \mathcal{A}_k|Pa(H_k, \mathcal{A}_k))$.

As mentioned earlier, such $\mathcal{L}_1$ and $\mathcal{L}_2$ distributional equivalence holds when the do-calculus rule-2 applies:

$$P_{Pa(H_k, \mathcal{A}_k)}(H_k, \mathcal{A}_k) = P(H_k, \mathcal{A}_k|Pa(H_k, \mathcal{A}_k)), \quad \text{if } (H_k, \mathcal{A}_k \perp\!\!\!\perp Pa(H_k, \mathcal{A}_k))_{G_{\underline{Pa(H_k, \mathcal{A}_k)}}} \tag{2}$$

This suggests a graphical criterion to find such a set $\mathcal{A}_k$ and we apply it at line 6 in Algorithm 1. Intuitively, if the outgoing edges of $Pa(H_k, \mathcal{A}_k)$ are deleted ($G_{\underline{Pa(H_k,\mathcal{A}_k)}}$) and they become d-separated from $\{H_k, \mathcal{A}_k\}$, then there exists no backdoor path from $Pa(H_k, \mathcal{A}_k)$ to $\{H_k, \mathcal{A}_k\}$ in G. Therefore, for a specific H_k, we start with $\mathcal{A}_k = \emptyset$ and check if $Pa(H_k, \mathcal{A}_k)$ satisfies the conditions of the rule-2 for $\{H_k, \mathcal{A}_k\}$. If not, we add parents of $\{H_k, \mathcal{A}_k\}$ to $\mathcal{A}_k$. We include ancestors, since only they can affect H_k's mechanisms from outside of the c-component. We continue the process until $Pa(H_k, \mathcal{A}_k)$ satisfies rule-2. Finally, finding a set $\mathcal{A}_k$ satisfying the modularity condition implies that we can train $\mathbb{G}_{H_k}$ by matching:

$$Q_{pa(H_k,\mathcal{A}_k)}(H_k, \mathcal{A}_k) = P(H_k, \mathcal{A}_k | Pa(H_k, \mathcal{A}_k)) \quad ; \text{Now training: } H_k, \text{Pre-trained: } \mathcal{A}_k \tag{3}$$

We utilize adversarial training to train the generators in $\mathbb{G}_{H_k}$ on observational dataset $\mathbf{D}$ to match the above. This is done by Algorithm 3: TrainModule() called in line 8. More precisely, this sub-routine uses all mechanisms in $\{H_k, \mathcal{A}_k\}$ to produce samples but only updates the mechanisms in $\mathbb{G}_{H_k}$ corresponding to the current h-node and returns those models after convergence. Even though we will train only $\mathbb{G}_{H_k}$ i.e., $\mathbb{G}_V, \forall V \in H_k$, $\mathcal{A}_k$ appears together with H_k in the joint distribution that we need to match. Thus, we use pre-trained causal mechanisms of $\mathcal{A}_k$, i.e., $\mathbb{G}_V, \forall V \in \mathcal{A}_k$ here. The partial order of $\mathcal{H}$-graph ensures that we have already trained $\mathcal{A}_k$ before H_k.

Training $\mathbb{G}_{H_k}$ to match the distribution in Equation 3, is sufficient to learn the c-factors $P_{Pa(C_j)}(C_j), \forall C_j \in H_k$. After training each $\mathbb{G}_{H_k}$ according to the partial order of $\mathcal{H}$-graph, Modular-DCM will learn a DCM that induces $Q(\mathcal{V}) = P(\mathcal{V})$. Finally, the trained DCM can sample from interventional $\mathcal{L}_2$ distributions identifiable from $P(\mathcal{V})$. These are formalized in Theorem 4.5. Proofs are in Appendix D.6.

Assumptions: 1. The true ADMG is known. 2. The causal model is semi-Markovian. 3. The data distribution is strictly positive. 4. Each conditional generative model $\mathbb{G}_i, \forall i$ in the DCM can correctly learn the target conditional distribution.

Theorem 4.5. *Consider any SCM* $\mathcal{M} = (G, \mathcal{N}, \mathcal{U}, \mathcal{F}, P(.))$. *Suppose Assumptions 1-4 hold. Algorithm 1 on* $(G, \mathbf{D})$ *returns a DCM* $\mathbb{G}$ *that entails* i) *the same observational distribution, and* ii) *the same identifiable interventional distributions as the SCM* $\mathcal{M}$.

5. Experimental Evaluation

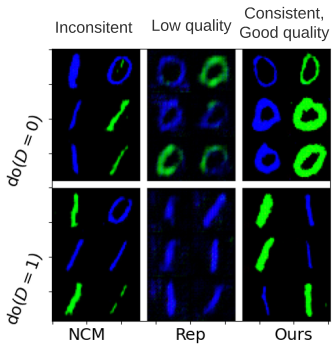
We present Modular-DCM performance on two semi-synthetic Colored-MNIST experiments and training convergence on a real-world COVIDx CXR-3 dataset provided in Appendix F.5. We also propose a solution to an invariant prediction problem for classification in CelebA-HQ. For distributions and image quality comparison, we use metrics such as the total variation distance (TVD), the KL-divergence, and the Frechet Inception Distance (FID). We share our implementation at https://github.com/Musfiqshohan/Modular-DCM.

5.1. Semi-Synthetic Colored-MNIST Experiments

MNIST frontdoor graph: We constructed a synthetic SCM that induces the graph in Figure 4a. Image variable I shows an image of the digit value of $Digit(D)$. We pick some random projection of the image as $Attribute(A)$ such that $P(A|do(D=0)) \neq P(A|do(D=1))$ holds, ensuring a strong causal effect. A hidden variable U affects both D and A such that $P(A|do(D)) \neq P(A|D)$. Suppose we are given a dataset $\mathcal{D}$ sampled from $P(D, A, I)$. Our goal is to estimate the causal effect $P(A|do(D))$. We can use the backdoor criterion (Pearl, 1993), to measure the ground truth $P(A|do(D)) = \int_U P(A|D, U)P(U)$.

To estimate $P(A|do(D))$ by training on the observational dataset $\mathcal{D}[D, A, I]$, we construct the Modular-DCM architecture with a neural network $\mathbb{G}_D$ having fully connected layers to produce D, a CNN-based generator $\mathbb{G}_I$ to generate images, and a classifier $\mathbb{G}_A$ to classify MNIST images into variable A such that D and A are confounded. Now, if we can train all mechanisms in the DCM to match $P(D, A, I)$, we can produce correct samples from $P(A|do(D))$. For this graph, the corresponding $\mathcal{H}$-graph is $[I] \rightarrow [D, A]$. Thus, we first train $\mathbb{G}_I$ by matching $P(I|D)$. Instead of training $\mathbb{G}_I$, we can also employ a pre-trained generative model that takes digits D as input and produces an MNIST image showing D digit in it. Next, we freeze $\mathbb{G}_I$ and train $\mathbb{G}_D$ and $\mathbb{G}_A$, to match the joint distribution $P(D, A, I)$ since $\{I\}$ is ancestor set $\mathcal{A}$ for c-component $\{D, A\}$. Convergence of generative models becomes difficult using the loss of this joint distribution since the losses generated by both low and high dimensional variables are non-trivial to compare and re-weight (see Appendix F.3). 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 $P(D, A, I)$.

Evaluation: In Figure 3, we compare our method with (Xia et al., 2023): NCM and a version of our method: DCM-Rep that does not use modular training (to serve as ablation study). First, we evaluate how each method matches the $\mathcal{L}_1$ and $\mathcal{L}_2$ distributions in Figure 3b. Since NCM trains all mechanisms with the same loss function involving both low and high-dimensional variables, it learns marginal distribution $P(I)$ but does not fully converge to match $P(A|\text{do}(D=0))$ finishing with TVD$=0.43$ at epoch 300. DCM-Rep uses a low-dim representation of images: RI and matches the joint distribution $P(D, A, RI)$ as a proxy to $P(D, A, I)$ without modularization. We observe DCM-Rep to converge slower (TVD$=0.36$ at epoch 300



(a) Generated Images

(b) TVD for frontdoor graph

Total Variation Distance (TVD) ↓	$P(D, A)$			$P(A\|\text{do}(D=0))$			$P(A\|\text{do}(D=1))$		
Epochs	25	150	**300**	25	150	**300**	25	150	**300**
NCM	0.14	0.16	0.17	0.48	0.42	0.43	0.08	0.11	0.11
DCM-Rep	0.38	0.17	0.16	0.36	0.45	0.36	0.22	0.1	0.11
DCM (Ours)	0.27	0.17	**0.08**	0.43	0.36	**0.13**	0.14	0.1	**0.04**

(c) FID for frontdoor graph

	Frechet Inception Distance (FID) ↓		
Epochs	25	150	**300**
NCM	61.40	59.28	59.59
NCM Pre-trained	26.60	184.40	192.27
DCM-Rep	151.41	78.69	80.65
DCM (Ours)	**27.02**	**27.37**	**27.20**

(d) FID for Diamond graph

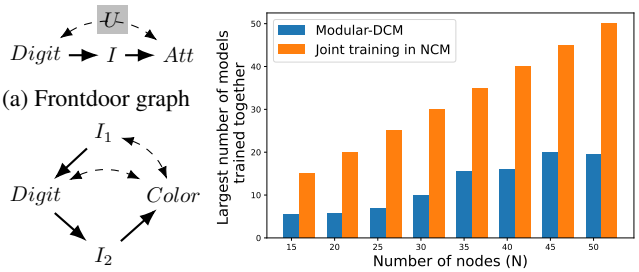
	Frechet Inception Distance FID(I_1) ↓		
Epochs	25	150	**300**
NCM	101.00	67.82	80.54
DCM (Ours)	79.96	35.74	**32.88**
	Frechet Inception Distance FID(I_2) ↓		
NCM	112.61	67.56	65.17
DCM (Ours)	22.22	15.47	**11.80**

Figure 3. For the frontdoor graph in Figure 4a, NCM produces good images but not consistent with do(D). Modular-DCM without modular training (DCM-rep) produces consistent but low-quality images. Our modular approach (DCM) with training order: $\{I\} \rightarrow \{D, A\}$ produces consistent, good images and converges faster (as shown in Figure 3a, 3b, 3c). In Figure 3d, we show our performance for the graph in Figure 4b that contains two image variables.

for $P(A|\text{do}(D=0))$) compared to the original Modular-DCM. Finally, Modular-DCM matches $P(D, A, RI)$ and converges faster with TVD$=0.08$ for $P(D, A)$ and 0.13 for $P(A|do(D=0))$. In Figure 3a, we show generated images of each method for do$(D=0)$(top) and do$(D=1)$(bottom) and evaluate their quality with FID scores in Figure 3c. We observe that NCM produces good-quality images (Figure 3a left, FID$=59.59$ at epoch 300) but is inconsistent with $do(D)$ intervention since it learns only marginal $P(I)$. DCM-Rep generates consistent but low-quality images (Figure 3a middle, FID$=80.65$). Modular-DCM equipped with a pre-trained model produces good-quality (FID$=27.20$), consistent $P(I|do(D))$ images (Figure 3a right). To justify the necessity of modularity, we add another method (NCM Pre-trained) in our ablation study, where we equip it with a pre-trained model but have to match the original joint by training all mechanisms together, the same as NCM. Note that it starts with a low FID (26.60) but ends up worsening the image quality with FID$=192.27$ (Figure 3c, row 2).

For a more rigorous evaluation, we use the effectiveness metric proposed in (Monteiro et al., 2023) and employ a classifier to map all images generated according to $P(I|\text{do}(D))$ back to discrete digits D. Next, we compute the exact likelihoods and compare with the true uniform intervention do$(D) : [0.5, 0.5]$ that we perform for $P(I|\text{do}(D))$. We observe that the results are consistent with Figure 3a. NCM generated images are classified as $[0.19, 0.81]$, implying that NCM learns only marginal $P(I)$. On the other hand, DCM-Rep and DCM generated images are classified as uniform distribution with 98% and 99% accuracy.

MNIST diamond graph: We illustrate our performance with a second synthetic SCM for the graph in Figure 4b. It contains multiple image nodes $\{I_1, I_2\}$ and discrete variables $Digit(D)$ and $Color(C)$. We consider a hidden con-



(a) Frontdoor graph

(b) Diamond graph

(c) Largest number of networks need to be trained together for different number of nodes.

Figure 4. Modular DCM on specific and arbitrary graphs.

founders between $\{I_1, C\}$ and one between $\{D, C\}$. Baseline NCM matches $P(I_1, D, I_2, C)$ by training mechanisms of all variables at the same time. Whereas we utilize the modularity offered by c-components $\{I_1, D, C\}$ and $\{I_2\}$. We i) first train I_2 to match $P(I_2|D)$ and then ii) train I_1, D, C to match $P(I_1, D, I_2, C)$ while freezing weights of I_2. At the first step, I_2 trains well. In the 2nd step, since we don't have to train I_2 anymore, we optimize a less complex loss function compared to NCM. In Figure 3d, we compare the FID scores of I_1 and I_2 generated by Modular-DCM and NCM. We observe that while both matches $P(D, C)$ (thus TVD omitted), DCM achieves FID$(I_1) = 32.88$ and FID$(I_2) = 11.80$ while NCM achieves FID$(I_1) = 80.54$ and FID$(I_2) = 65.17$ after running for 300 epochs.

Arbitrary graphs: In this experiment, we showcase the benefit of modularization over a random ensemble of graphs. We numerically visualize the largest number of mechanisms we have to update and train together compared to the full training of existing works. We sample random DAGs with a varying number of nodes ($N \in [15 - 50]$) keeping the arc ratio and the number of latents equal to $N/3$. We call Al-

gorithm 7: Construct_H-graph(.) to find the largest training component. We took the average of five runs and plot it in Figure 4c. This plot demonstrates the number of networks that are trained together in a single training phase.

We observe that the number of models NCM trains together increases linearly with respect to N whereas the growth in our method is relatively smaller since it does not depend on the number of nodes, but rather on the number of latents. For a graph of $N = 50$ variables, NCM updates the 50 networks corresponding to the mechanisms of all variables with a common loss function. Whereas, we train the same set of neural networks but modularly c-component by c-component with average max size of 20 (for the setting in Figure 4c). We achieve better convergence since we minimize a less complex loss function at each training phase. We experience such convergence for Colored-MNIST with a low total variation distance compared to NCM (Figure 3c). We discuss complexity evaluation of our algorithm in Appendix F.2.

5.2. Invariant Prediction on CelebA-HQ

In this section, we design a causal invariant classifier f for the high-dimensional image dataset, CelebA-HQ (Lee et al., 2020) such that its specific attribute classification $Eyeglass = f(Image)$; does not experience low accuracy with domain shift.

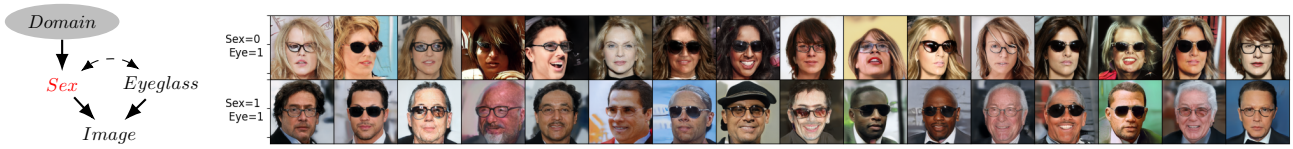
Motivation: Among all attributes of CelebA-HQ, some attributes, such as Sex and $Eyeglass$ have spurious correlations between them (Shen et al., 2020) (men are more likely to wear eyeglasses, correlation coefficient 0.47). A classifier trained on this dataset might consider the facial features of a male as an indicator to predict the presence of eyeglasses. As a result, if there is a shift in the sex distribution in the test domain, i.e., $P(Sex|domain = test) \neq P(Sex|domain = train)$ (Figure 5a), and the classifier has to predict more images of females, it might have low eyeglass accuracy. For example, in Figure 5b (row-1), the accuracy for such a classifier in the $Sex = 0, Eyeglass = 1$ sub-population is 0.81 (comparatively lower). We model the above scenario with the causal graph in Figure 5a (top-left). We assume Eyeglass and Sex attributes determine how the Image variable would look like (shown with directed edges). The spurious correlation between the attributes is represented with a bi-directed edge. We reflect the distribution shift in $P(Sex)$, with an edge from the $Domain$ variable.

To make the CelebA-HQ attribute classification independent of the domain shift, we employ causal invariant prediction. Causal invariant prediction refers to the problem of learning a predictive model which is invariant to specific distribution shifts. According to (Subbaswamy et al., 2019; Lee et al., 2023), to build a causal invariant predictor, we need to train it on an interventional dataset where the target attribute is independent of spuriously correlated/sensitive attributes due to the intervention performed. In our context, we need interventional samples from the high-dimensional interventional distribution $P(Eyeglass, Image|\text{do}(Sex))$ to train the invariant classifier, since the connection between $Domain$ and $Eyeglass$ is cut off by $\text{do}(Sex)$. The first step to obtain these interventional samples is to train a deep causal generative model and learn the observational distribution $P(Sex, Eyeglass, Image)$. For this purpose, we utilize our algorithm Modular-DCM.

Dataset: The original CelebA-HQ dataset contains 1468 images of $Eyeglasses = 1$. We distribute these samples among 5380 train samples and 1280 test samples maintaining the joint distribution in Figure 5a such that the distribution shift in $P(Sex)$ is reflected across domains.

Training: Here we discuss three classifiers that we trained. `TrainDomain`: This classifier is trained on the training dataset. `Intervention`: According to (Subbaswamy et al., 2019), if we can generate a dataset $\mathcal{D}[Eyeglass, Image] \sim P(Eyeglass, Image|\text{do}(Sex))$ and train a classifier on this dataset, its prediction will be invariant to the $Domain$ and Sex, since intervention on Sex removes their influence. To generate this high-dimensional interventional dataset, Modular-DCM employs neural networks $(\mathbb{G}_{Eyeglass}, \mathbb{G}_{Sex}, \mathbb{G}_I)$ for each of $Eyeglass, Sex$ and $Image$ and connects them according to the causal graph in Figure 5a. First, we train $\{\mathbb{G}_{Eyeglass}, \mathbb{G}_{Sex}\}$ together (same c-component). Now, for $\mathbb{G}_I$, Modular-DCM's flexibility to incorporate pre-trained networks in its causal generative models allows us to utilize InterFaceGAN (Shen et al., 2020), that uses StyleGAN (Karras et al., 2019) under the hood to produce realistic human faces. This pre-trained model plays an important role for a classifier since it needs to see realistic images and training a model from scratch to match the true $P(I|Eyeglass, Sex)$ will be costly. Next, we uniformly intervene on $Sex = 0$ and $Sex = 1$ and push forward through the trained models to generate 10k samples $\mathcal{D} \sim P(Eyeglass, Image|\text{do}(Sex))$ of both females and males. Finally, we train a new classifier `Intervention` on this dataset for eyeglass prediction. `Augmented` : To obtain the benefits of both classifiers, we create an augmented dataset by combining the training dataset and the interventional dataset generated by Modular-DCM. We train a new classifier `Augmented` on it.

Evaluation: We evaluate the accuracy of the classifiers in the test domain (Figure 5b). Since `TrainDomain` (row-1) learns the $Male\text{-}Eyeglass$ bias, it achieves accuracy= 0.90 in the $Sex = 1, Eyeglass = 1$ sub-population but it performs badly for $Sex = 0, Eyeglass = 1$ (accuracy 0.81). `Intervention` (row-3) is trained on images generated by InterFaceGAN, but due to the large support of the im-



	Train	
	EG=0	EG=1
Sex=0	0.60	0.018
Sex=1	0.18	0.20
	Test	
Sex=0	0.31	0.14
Sex=1	0.47	0.08

(a) Train & Test distribution

Classifier/ Accuracy ↑	Sex=0	Sex=1	Eyeglass=0	Eyeglass=1	Sex=0 Eyeglass=0	Sex=0 Eyeglass=1	Sex=1 Eyeglass=0	Sex=1 Eyeglass=1
`TrainDomain`	0.948	0.946	0.971	0.847	0.998	0.810	0.953	0.90
`IRM`	0.880	0.866	0.918	0.681	0.980	0.600	0.877	0.800
`Intvention (Ours)`	0.913	0.944	0.993	0.674	1.000	0.669	0.988	0.68
`Augmented (Ours)`	0.970	0.981	**0.994**	**0.905**	0.995	**0.901**	0.993	0.91

(b) Image samples and classifier accuracy for different sub-population

Figure 5. (Top-Left): Invariant prediction causal graph. (Top-right) Images generated by InterFaceGAN from $P(I|Sex, Eyeglass = 1)$. (Bottom-left): Joint distribution of $P(Sex, Eyeglass)$. (Bottom-right): Eyeglass prediction accuracy of 3 classifiers in different sub-populations. Three classifiers are trained on the training dataset, the interventional dataset, and the augmented dataset (combined both). Note that the `Augmented` has better accuracy in the $Sex = 0, Eyeglass = 1$ sub-population which was our target to achieve.

age manifold, samples generated by InterFaceGAN from $P(Image|Eyeglass, Sex)$ might not represent all types of images that are present in the original CelebA-HQ dataset. For example, CelebA-HQ contains more variety of sunglasses compared to the InterFaceGAN generated images (Figure 22 vs 21). As a result, `Intervention` does not perform very well in the $Eyeglass = 1$ sub-population (0.67). However, the generated interventional dataset contains images of both $Sex = 0, 1$ wearing eyeglasses which are free from the training dataset bias. Thus, when we combine both datasets, samples from the training dataset introduce the `Augmented` classifier (row-4) to different varieties of $Eyeglass = 1$ images and samples from the interventional dataset enforce it to focus on only eyeglass property in an image. We observe that `Augmented` improves accuracy in the three sub-populations (bolded in Figure 5b) : i)$\{Sex = 0\} : 0.948 \rightarrow 0.97$ ii)$\{Eyeglass = 1\} : 0.847 \rightarrow 0.905$ and iii)$\{Sex = 0, Eyeglass = 1\} : 0.810 \rightarrow 0.901$. Thus, even though we have access to only the biased observational dataset, Modular-DCM offered a bias-free interventional dataset and enabled us to train a domain invariant classifier.

Note that we also evaluate the Invariant Risk Minimization (IRM) (Arjovsky et al., 2019) method for this specific experiment although the problem setup of IRM and our proposed method are different. We train IRM on data from two environments: $Sex = 0$ and $Sex = 1$. We provide its accuracy in Figure 5b (row-2) and show that the classifiers trained according to our approach outperform it in most cases.

6. Conclusion

We propose a modular adversarial training algorithm for learning deep causal generative models and estimate causal effects with high-dimensional variables in the presence of confounders. After convergence, Modular-DCM can generate high-dimensional samples from identifiable interventional distributions. We assume the causal model to be semi-Markovian which aim to relax in our future work. Some potential application of our algorithm includes: continual learning (Busch et al., 2023) of deep causal generative models or high-dimensional interventional sampling in federated setting (Vo et al., 2022).

Acknowledgements

This research has been supported in part by NSF CAREER 2239375, IIS 2348717, Amazon Research Award and Adobe Research.

Impact Statement

Our proposed algorithm Modular-DCM, can sample from high-dimensional observational and interventional distributions. As a result, it can be used to explore different creative directions such as producing realistic interventional images that we can not observe in real world. We can train Modular-DCM models on datasets and perform an intervention on sensitive attributes to detect any bias towards them or any unfairness against them (Xu et al., 2019; van Breugel et al., 2021). However, an adversary might apply our method to produce realistic images that are causal. As a result, it will be harder to detect fake data generated by DCM.

References

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv*, 2019.

Balazadeh Meresht, V., Syrgkanis, V., and Krishnan, R. G. Partial identification of treatment effects with implicit generative models. *Advances in Neural Information Processing Systems*, 35:22816–22829, 2022.

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, 2012a. AUAI Press. ISBN 9780974903989.

Bareinboim, E. and Pearl, J. Causal inference by surrogate experiments: z-identifiability. *arXiv preprint arXiv:1210.4842*, 2012b.

Bica, I., Jordon, J., and van der Schaar, M. Estimating the effects of continuous-valued interventions using generative adversarial networks. *Advances in Neural Information Processing Systems*, 33:16434–16445, 2020.

Busch, F. P., Seng, J., Willig, M., and Zečević, M. Continually updating neural causal models. In *AAAI Bridge Program on Continual Causality*, pp. 30–37. PMLR, 2023.

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

Chao, P., Blöbaum, P., and Kasiviswanathan, S. P. Interventional and counterfactual inference with diffusion models. *arXiv preprint arXiv:2302.00860*, 2023.

Dash, S., Balasubramanian, V. N., and Sharma, A. Evaluating and mitigating bias in image classifiers: A causal perspective using counterfactuals. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 915–924, 2022.

Gao, C., Zheng, Y., Wang, W., Feng, F., He, X., and Li, Y. Causal inference in recommender systems: A survey and future directions. *arXiv preprint arXiv:2208.12397*, 2022.

Giorgio Carbone, Remo Marconzini, G. C. Cxr-acgan: Auxiliary classifier gan (ac-gan) for conditional generation of chest x-ray images (pneumonia, covid-19 and healthy patients) for the purpose of data augmentation. https://github.com/giocoal/CXR-ACGAN-chest-xray-generator-covid19-pneumonia, 2023.

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.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Huang, Y. and Valtorta, M. Pearl's calculus of intervention is complete. *arXiv preprint arXiv:1206.6831*, 2012.

Jaber, A., Zhang, J., and Bareinboim, E. Causal identification under markov equivalence: Completeness results. In *International Conference on Machine Learning*, pp. 2981–2989. PMLR, 2019.

Jerzak, C. T., Johansson, F., and Daoud, A. Image-based treatment effect heterogeneity. *arXiv preprint arXiv:2206.06417*, 2022.

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.

Lee, C.-H., Liu, Z., Wu, L., and Luo, P. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Lee, K., Rahman, M. M., and Kocaoglu, M. Finding invariant predictors efficiently via causal structure. In *Uncertainty in Artificial Intelligence*, pp. 1196–1206. PMLR, 2023.

Louizos, C., Shalit, U., Mooij, J. M., Sontag, D., Zemel, R., and Welling, M. Causal effect inference with deep latent-variable models. *Advances in neural information processing systems*, 30, 2017.

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

Monteiro, M., Ribeiro, F. D. S., Pawlowski, N., Castro, D. C., and Glocker, B. Measuring axiomatic soundness of counterfactual image models. *arXiv preprint arXiv:2303.01274*, 2023.

Nemirovsky, D., Thiebaut, N., Xu, Y., and Gupta, A. Countergan: generating realistic counterfactuals with residual generative adversarial nets. *arXiv preprint arXiv:2009.05199*, 2020.

Pawlowski, N., Coelho de Castro, D., and Glocker, B. Deep structural causal models for tractable counterfactual inference. *Advances in Neural Information Processing Systems*, 33:857–869, 2020.

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.

Qin, W., Zhang, H., Hong, R., Lim, E.-P., and Sun, Q. Causal interventional training for image recognition. *IEEE Transactions on Multimedia*, 2021.

Ribeiro, F. D. S., Xia, T., Monteiro, M., Pawlowski, N., and Glocker, B. High fidelity image counterfactuals with probabilistic causal models. *arXiv preprint arXiv:2306.15764*, 2023.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., and Nolan, G. P. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721): 523–529, 2005.

Scutari, M. and Denis, J.-B. *Bayesian networks: with examples in R*. Chapman and Hall/CRC, 2021.

Seitzer, M. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, August 2020. Version 0.3.0.

Shalit, U., Johansson, F. D., and Sontag, D. Estimating individual treatment effect: generalization bounds and algorithms. In *International conference on machine learning*, pp. 3076–3085. PMLR, 2017.

Shen, Y., Gu, J., Tang, X., and Zhou, B. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9243–9252, 2020.

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.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Subbaswamy, A., Schulam, P., and Saria, S. Preventing failures due to dataset shift: Learning predictive models that transport. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 3118–3127. PMLR, 2019.

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

Tian, J., Kang, C., and Pearl, J. A characterization of interventional distributions in semi-markovian causal models. In *AAAI*, pp. 1239–1244, 2006.

van Breugel, B., Kyono, T., Berrevoets, J., and van der Schaar, M. Decaf: Generating fair synthetic data using causally-aware generative networks. *Advances in Neural Information Processing Systems*, 34:22221–22233, 2021.

Vo, T. V., Bhattacharyya, A., Lee, Y., and Leong, T.-Y. An adaptive kernel approach to federated learning of heterogeneous causal effects. *Advances in Neural Information Processing Systems*, 35:24459–24473, 2022.

Wang, L., Lin, Z. Q., and Wong, A. Covid-net: a tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. *Scientific Reports*, 10(1):19549, Nov 2020. ISSN 2045-2322. doi: 10.1038/s41598-020-76550-z. URL https://doi.org/10.1038/s41598-020-76550-z.

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.

Xu, D., Wu, Y., Yuan, S., Zhang, L., and Wu, X. Achieving causal fairness through generative adversarial networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

Yang, M., Liu, F., Chen, Z., Shen, X., Hao, J., and Wang, J. Causalvae: Disentangled representation learning via neural structural causal models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9593–9602, 2021.

Zhang, W., Liu, L., and Li, J. Treatment effect estimation with disentangled latent factors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10923–10930, 2021.

A. 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. We also assume each confounder to cause only two observed variables, which is considered as semi-Markovian in the literature. Another limitation of our work is that same as our close baselines, we do not consider conditional sampling. Modular-DCM can perform rejection sampling, which is practical if the evidence variables are low-dimensional. 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. We aim to extend our work for non-Markovian causal models where confounders can cause any number of observed variables. We aim to resolve these limitations in our future work.

B. Modular-DCM Training on Interventional Datasets

Although Theorem 4.5 focuses training on observational data and sampling from interventional distributions, it can trivially be generalized to z-identifiability (Bareinboim & Pearl, 2012b). That is we can generate samples from other interventional distributions that are non-identifiable from only observational data but are identifiable from a combination of both observational and interventional data. This can be trivially done by expanding the notion of identifiability to use a given collection of interventional distributions, and requiring Modular-DCM to entail the same interventional distributions for the said collection. Thus in the appendix we provide proofs for both setups.

C. Appendix: Modular-DCM: Adversarial Training of Deep Causal Generative Models

Definition C.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 C.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, 2012a) proposes a z-identification algorithm to derive g_G for these ϕ and θ

C.1. Modular-DCM Interventional Sampling after Training

After Modular-DCM training, to perform hard intervention and produce samples accordingly, we manually set values of the intervened variables instead of using their neural network. Then, we feed forward those values into its children's mechanisms and generate rest of the variable like as usual. Figure 6(b) is the Modular-DCM network for the causal graph in Figure 6(a). Now, in Figure 6(c), we performed do$(X = x)$. Exogenous variables U_1 and n_X are not affecting X anymore as we manually set $X = x$.

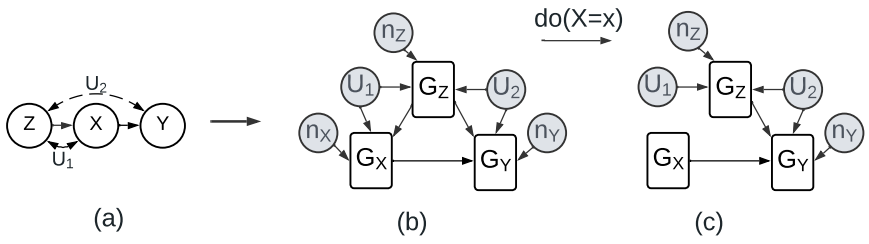


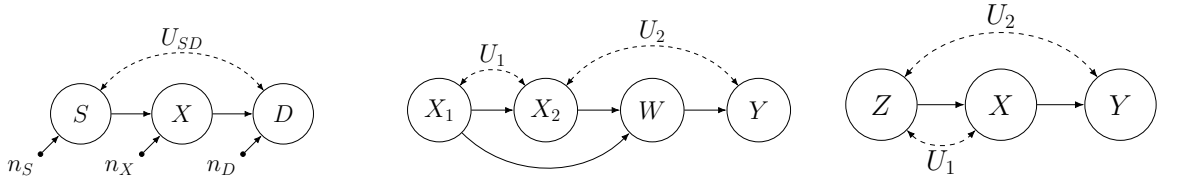
Figure 6. (a) Causal Graph with latents. (b), (c) DCM before and after intervention.

C.2. Modular-DCM: Adversarial Training of Deep Causal Generative Models (Full Training)

Full training of the DCM indicates the setup when we update all mechanisms in the causal graph with the same common loss. In this section, we prove that a trained DCM can sample from identifiable causal queries from any causal layer. We assume M_1 as true SCM and M_2 as DCM of Modular-DCM.

Theorem C.3. *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 and/or interventional distributions $\{P_i(\mathcal{V})\}_{i\in[m]}$ 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 $\{P_i(\mathcal{V})\}_{i\in[m]} \vdash \mathcal{K}_{\mathcal{M}_1}(\mathcal{V})$ and $P_i(\mathcal{V}) = Q_i(\mathcal{V}), \forall i \in [m]$, then $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V}) = \mathcal{K}_{\mathcal{M}_2}(\mathcal{V})$.*

Proof. By definition of identifiability, we have that $\mathcal{K}_{\mathcal{M}_1} = g_G(\{P_i(\mathcal{V})\}_{i\in[m]})$ 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., $\mathcal{K}_{\mathcal{M}_2} = g_G(\{Q_i(\mathcal{V})_{i\in[m]}\})$. Thus, the query has the same answer in both SCMs, if they entail the same input distributions over the observed variables, i.e., $P_i(\mathcal{V}) = Q_i(\mathcal{V}), \forall i$. □



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

Corollary C.4. *Let $\mathcal{M}_1 = (G = (\mathcal{V}, \mathcal{E}), \mathcal{N}, \mathcal{U}, \mathcal{F}, P(.))$ and $\mathcal{M}_2 = (G, \mathcal{N}', \mathcal{U}', \mathcal{F}', Q(.))$ be two SCMs. If $\{P(\mathcal{V})\} \vdash P_x(Y)$ for $X, Y \subset \mathcal{V}$, $X \cap Y = \emptyset$ and $P(\mathcal{V}) = Q(\mathcal{V})$ then $P_x(Y) = Q_x(Y)$*

For example, in Figure 7(b), the interventional query $P_{x_1,x_2}(W)$ is identifiable from $P(\mathcal{V})$. According to the Corollary C.4, after training on $P(\mathcal{V})$ dataset, Modular-DCM will produce correct interventional sample from $P_{x_1,x_2}(W)$ and along with other queries in $\mathcal{L}_2(P(\mathcal{V}))$.

(Bareinboim & Pearl, 2012b) showed that we can identify some $\mathcal{L}_2$-queries with other surrogate interventions and $\mathcal{L}_1$-distributions. Similarly, we can apply Theorem C.3:

Corollary C.5. *Let $\mathcal{M}_1 = (G = (\mathcal{V}, \mathcal{E}), \mathcal{N}, \mathcal{U}, \mathcal{F}, P(.))$ and $\mathcal{M}_2 = (G, \mathcal{N}', \mathcal{U}', \mathcal{F}', Q(.))$ be two SCMs and X, Y be disjoint, and $\{S_i\}_i$ arbitrary subsets of variables. If $i)\{P(\mathcal{V}), P_{s_1}(\mathcal{V}), P_{s_2}(\mathcal{V}) \ldots\} \vdash P_x(Y)$, $ii) P(\mathcal{V}) = Q(\mathcal{V})$ and $iii) P_{s_i}(\mathcal{V}) = Q_{s_i}(\mathcal{V})$, $\forall i, s_i$ then $P_x(Y) = Q_x(Y)$.*

In Figure 7(c), the interventional query $P_x(Y)$ is identifiable from $P(\mathcal{V})$ and $P_z(X, Y)$. Therefore, after being trained on datasets sampled from these distributions, Modular-DCM will produce correct interventional sample from $P_x(Y)$ and all other queries in $\mathcal{L}_2(P(\mathcal{V}), P_z(X, Y))$.

C.3. Training with Multiple Datasets

We propose a method in Algorithm 2 for training Modular-DCM with both $\mathcal{L}_1$ and $\mathcal{L}_2$ datasets. We use Wasserstein GAN with penalized gradients (WGAN-GP) (Gulrajani et al., 2017) for adversarial training. We can also use more recent generative models such as diffusion models when variables are not in any c-component. $\mathbb{G}$ is the DCM, a set of generators and $\{\mathbb{D}_x\}_{X\in\mathbf{I}}$ are a set of discriminators for each intervention value combinations. The objective function of a two-player minimax game would be

$$\min_{\mathbb{G}} \sum_{x} \max_{\mathbb{D}_x} L(\mathbb{D}_x, \mathbb{G}),$$

$$L(\mathbb{D}_x, \mathbb{G}) = \mathop{\mathbb{E}}_{v\sim\mathbb{P}_x^r}[\mathbb{D}_x(\mathbf{v})] - \mathop{\mathbb{E}}_{\mathbf{z}\sim\mathbb{P}_Z, \mathbf{u}\sim\mathbb{P}_U}[\mathbb{D}_x(\mathbb{G}^{(\mathbf{x})}(\mathbf{z}, \mathbf{u}))]$$

Algorithm 2 Modular-DCM Training on Multiple Datasets

1: **Input:** Causal Graph $G = (\mathcal{V}, \mathcal{E})$, Interventional datasets= $(\mathbf{I}, \mathcal{D})$, DCM $\mathbb{G}$, Critic $\mathbb{D}$, Parameters= $\theta_1, \ldots, \theta_n$, $\lambda = 10$
2: **while** $\theta_1, \ldots, \theta_n$ has not converged **do**
3: **for each** $(X, D) \in (\mathbf{I}, \mathcal{D})$ **do**
4: $compare\_var = \mathcal{V}$
5: Sample real data $\mathbf{v}^{\mathbf{r}}_{\mathbf{x}} \sim D$ following the distribution $\mathbb{P}^r_x$ with intervention X.
6: $\mathbf{x} \leftarrow X.values$ $\quad$ *//X=(keys, values)*
7: $\mathbf{v}^f_x = \text{RunGAN}(G, \mathbb{G}, X, compare\_var, \emptyset)$
8: $\hat{\mathbf{v}}_x = \epsilon \mathbf{v}^r_x + (1-\epsilon)\mathbf{v}^f_x$
9: $L_x = \mathbb{D}_{w_x}(\mathbf{v}^f_x) - \mathbb{D}_{w_x}(\mathbf{v}^r_x) * \lambda(\|\nabla_{\hat{\mathbf{v}}_x} \mathbb{D}_{w_x}(\hat{\mathbf{v}}_x)\|_2 - 1)^2$
10: $G_{loss} = G_{loss} + \mathbb{D}_{w_x}(\mathbf{v}^f_x)$
11: $w_x = Adam(\nabla_{w_x} \frac{1}{m} \sum_{j=1}^{m} L_{\mathbb{D}_x}, w_x, \alpha, \beta_1, \beta_2)$
12: **for** $\theta \in \theta_1, \ldots \theta_n$ **do**
13: $\theta = Adam(\nabla_\theta - G_{loss}, \theta, \alpha, \beta_1, \beta_2)$
14: **Return:** $\theta_1, \ldots \theta_n$

Here, for intervention $\text{do}(X = x), X \in \mathbf{I}$, $\mathbb{G}^{(x)}(\mathbf{z}, \mathbf{u})$ are generated samples and $v \sim \mathbb{P}^r_x$ are real $\mathcal{L}_1$ or $\mathcal{L}_2$ samples. We train our models by iterating over all datasets and learn $\mathcal{L}_1$ and $\mathcal{L}_2$ distributions (line 3). We produce generated interventional samples by intervening on the corresponding node of our architecture. For this purpose, we call Algorithm 9 RunGAN(), at line 7. We compare the generated samples with the input $\mathcal{L}_1$ or $\mathcal{L}_2$ datasets. For each different combination of the intervened variables x, $\mathbb{D}_x$ will have different losses, $L_{X=x}$ from each discriminator (line 9). At line 10, we calculate and accumulate the generator loss over each dataset. If we have $V_1, \ldots, V_n \in \mathcal{V}$, then we update each variable's model weights based on the accumulated loss (line 13). This will ensure that after convergence, Modular-DCM models will learn distributions of all the available datasets and according to Theorem C.3, it will be able to produce samples from same or higher causal layers queries that are identifiable from these input distributions. Following this approach, Modular-DCM Training in Algorithm 2, will find a DCM solution that matches to all the input distributions, mimicking the true SCM. Finally, we describe sampling method for Modular-DCM after training convergence in Appendix C.1.

Proposition C.6. *Let $\mathcal{M}_1$ be the true SCM and Algorithm 2: **Modular-DCM Training** converges after being trained on datasets: $\mathbf{D} = \{\mathcal{D}_i\}_i$, outputs the DCM $\mathcal{M}_2$. If for any causal query $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V})$ identifiable from $\mathbf{D}$ then $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V}) = \mathcal{K}_{\mathcal{M}2}(\mathcal{V})$*

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 Modular-DCM. Modular-DCM Training converges implies that $Q_i(\mathcal{V}) = P_i(\mathcal{V}), \forall i \in [m]$ for all input distributions. Therefore, according to Theorem C.3, Modular-DCM is capable of producing samples from correct interventional distributions that are identifiable from the input distributions. □

C.4. Non-Markovianity

Note that, one can convert a non-Markovian causal model M_1 to a semi-Markovian causal model M_2 by taking the common confounder among the observed variables and splitting it into new confounders for each pair. Now, for a causal query to be unidentifiable in a semi-Markovian model M_2, we can apply the Identification algorithm (Shpitser & Pearl, 2008) and check if there exists a hedge. The unidentifiability of the causal query does not depend on the confounder distribution. Thus, if the causal query is unidentifiable in the transformed semi-Markovian model M_2, it will be unidentifiable in the original non-Markovian model M_1 as well.

Besides Semi-Markovian, Theorem 4.2 and Theorem C.3 holds for Non-Markovian models, with latents appearing anywhere in the graph and thus can be learned by Modular-DCM training. (Jaber et al., 2019) performs causal effect identification on equivalence class of causal diagrams, a partial ancestral graph (PAG) that can be learned from observational data. Therefore, we can apply their method to check if an interventional query is identifiable from observational data in a Non-Markovian causal model and express the query in terms of observations and obtain the same result as Theorem C.3. We aim to explore these directions in more detail in our future work.

D. Appendix: Modular-DCM Modular Training

D.1. Tian's Factorization for Modular Training

In Figure 8(a), We apply Tian's factorization (Tian & Pearl, 2002) to get,

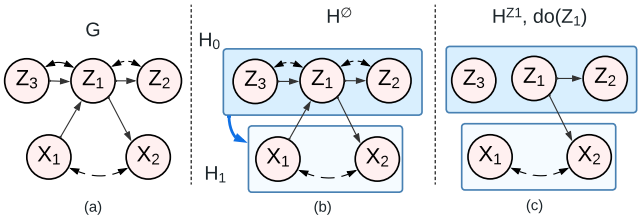


Figure 8. (a) Causal graph G, (b) $\mathcal{H}^{\emptyset}$-graph, (c) H^{Z_1}-graph

$$P(v) = P(x_1, x_2|\text{do}(z_1))P(z_1, z_2, z_3|\text{do}(x_1)) \tag{4}$$

We need to match the following distributions with the DCM.

$$\begin{aligned} P(x_1, x_2|\text{do}(z_1)) &= Q(x_1, x_2|\text{do}(z_1)) \\ P(z_1, z_2, z_3|\text{do}(x_1)) &= Q(z_1, z_2, z_3|\text{do}(x_1)) \end{aligned} \tag{5}$$

With modular training, we matched the following alternative distributions:

$$\begin{aligned} P(z_1, z_2, z_3|x_1) &= Q(z_1, z_2, z_3|\text{do}(x_1)) \\ P(x_1, x_2, z_1, z_3) &= Q(x_1, x_2, z_1, z_3) \end{aligned} \tag{6}$$

Now, for the graph in Figure 8(a),

$$\begin{aligned} P(x_1, x_2, z_1, z_2, z_3) &= P(x_1, x_2|\text{do}(z_1)) \times P(z_1, z_2, z_3|\text{do}(x_1) \\ &= \frac{P(x_1, x_2, z_1, z_3)}{P(z_1, z_3|\text{do}(x_1))} \times P(z_1, z_2, z_3|x_1) \quad \text{[C-factorization of } P(x_1, x_2, z_1, z_3)\text{]} \\ &= \frac{P(x_1, x_2, z_1, z_3)}{P(z_1, z_3|x_1)} \times P(z_1, z_2, z_3|x_1) \quad \text{[Do-calculus rule-2 applies]} \\ &= \frac{P(x_1, x_2, z_1, z_3)}{\sum_{z_2} P(z_1, z_2, z_3|x_1)} \times P(z_1, z_2, z_3|x_1) \\ &= \frac{Q(x_1, x_2, z_1, z_3)}{\sum_{z_2} Q_{x_1}(z_1, z_2, z_3)} \times Q_{x_1}(z_1, z_2, z_3) \quad \text{[According to Equation 6]} \\ &= Q(x_1, x_2, z_1, z_2) \quad \text{[We can follow the same above steps as } P(.) \text{ for } Q(.)\text{]} \end{aligned} \tag{7}$$

Therefore, if we match the distributions in Equation 6 with the DCM, it will match $P(\mathcal{V})$ as well.

D.2. Modular Training for Interventional Dataset

D.2.1. Modular Training Basics

Suppose, for the graph in Figure 9, we have two datasets $D^{\emptyset} \sim P(\mathcal{V})$ and $D^{z_1} \sim P_{Z_1}(V)$, i.e., intervention set $\mathcal{I} = \{\emptyset, Z_1\}$. Joint distributions in both dataset factorize like below:

$$\begin{aligned} P(v) &= P_{z_1}(x_1, x_2)P_{x_1}(z_1, z_2, z_3) \\ P_{z_1}(v) &= P_{z_1}(x_1, x_2)P(z_3)P_{z_1}(z_2) \\ &= P_{z_1}(x_1, x_2)P_{z_1}(z_2, z_3) \text{[Since } Z_2, Z_3 \text{ independent in } G_{\overline{Z_1}} \text{ graph]} \\ &= P_{z_1}(x_1, x_2)P_{x_1, z_1}(z_2, z_3) \text{[Ignores intervention using do calculus rule-2]} \end{aligned} \tag{8}$$

We change the c-factors for $P_{z_1}(V)$ to keep the variables in each c-factor same in all distributions. This factorization suggests that to match $P(\mathcal{V})$ and $P_{Z_1}(V)$ we have to match each of the c-factors using $D^{\emptyset}$ and D^{z_1} datasets. In Figure 2a graph G, $P_{x_1}(z_1, z_2, z_3) = P(z_1, z_2, z_3|x_1)$ since do-calculus rule-2 applies. And in $G_{\overline{Z_1}}$, $P(z_3)P_{z_1}(z_2)$ can be combined into $P_{x_1, z_1}(z_2, z_3)$. Thus we can use these distributions to train part of the DCM: $\mathbb{G}_{Z_1}, \mathbb{G}_{Z_2}, \mathbb{G}_{Z_3}$ to learn

both $Q(z_1, z_2, z_3|\text{do}(x_1))$ and $Q(z_2, z_3|\text{do}(x_1, z_1))$. However, $P(x_1, x_2|\text{do}(z_1)) \neq P(x_1, x_2|z_1)$ in $P(\mathcal{V})$. But we have access to $P_{z_1}(\mathcal{V})$. Thus, we can train $\mathbb{G}_{X_1}, \mathbb{G}_{X_2}$ with only dataset $D^{Z_1} \sim P_{z_1}(\mathcal{V})$ (instead of both $D^{\emptyset}, D^{Z_1}$) and learn $Q(x_1, x_2|\text{do}(z_1))$. This will ensure the DCM has matched both $P(\mathcal{V})$ and $P_{z_1}(\mathcal{V})$ distribution.

Thus, we search proxy distributions to each c-factor corresponding to both $P(\mathcal{V})$ and $P_{Z_1}(V)$ dataset, to train the mechanisms in a c-component $\mathbf{Y}$. For each of the c-factors corresponding to $\mathbf{Y}$ in $P(\mathcal{V})$ and $P_{z_1}(V)$, we search for two ancestor sets $\mathcal{A}_{\emptyset}, \mathcal{A}_{Z_1}$ in both $P(\mathcal{V})$ and $P_{z_1}(V)$ datasets such that the parent set $Pa(\mathbf{Y} \cup \mathcal{A}_{\emptyset})$ satisfies rule-2 for the joint $\mathbf{Y} \cup \mathcal{A}_{\emptyset}$ and $Pa(\mathbf{Y} \cup \mathcal{A}_{Z_1})$ satisfies rule-2 for the joint $\mathbf{Y} \cup \mathcal{A}_{Z_1}$ with $\text{do}(Z_1)$ intervention.

We update Definition 4.4as **modularity condition-I** for multiple interventional datasets as below:

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

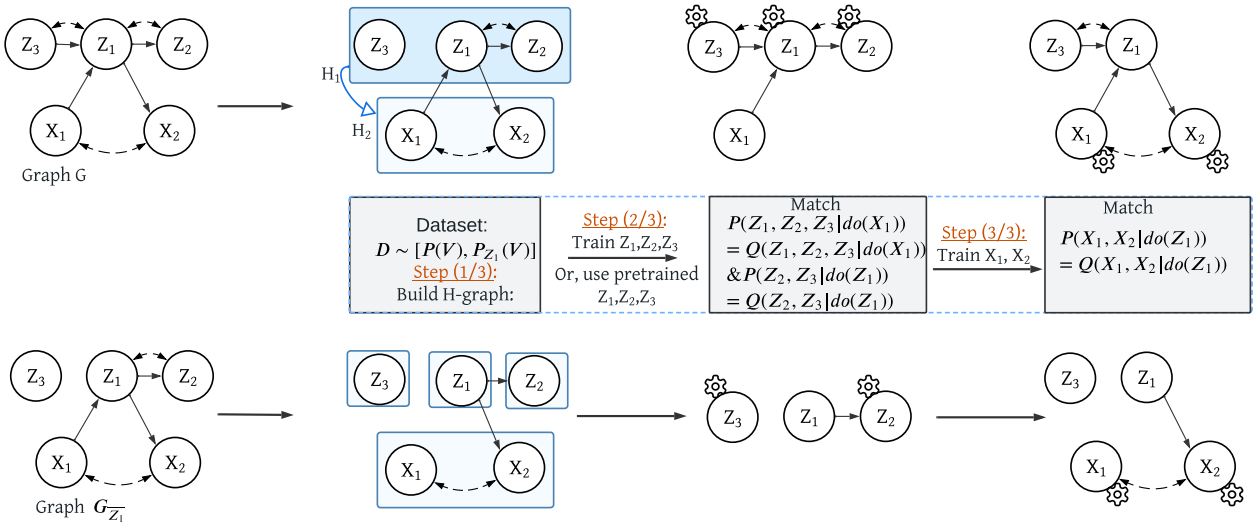


Figure 9. Modular training on H-graph: $H_1 : [Z_1, Z_2, Z_3] \rightarrow H_2 : [X_1, X_2]$ with dataset $D \sim P(V)$.

Unlike before, we have access to $P_{z_1}(\mathcal{V})$ and we can use that to match $P_{z_1}(x_1, x_2)$ in both cases. To match the $\mathcal{L}_1$ and $\mathcal{L}_2$ joint distributions according to (8), we train each c-component one by one. For each c-component, we identify the modularity conditions of all c-factors $P_{pa(\mathbf{Y})\cup I}(\mathbf{Y}), \forall I \in \mathcal{I}$ and use them to train $\mathbf{Y}$. We train the mechanisms in $\mathbf{Y}$ to learn an alternative to each c-factor $P_{pa(\mathbf{Y})\cup I}(\mathbf{Y}), \forall I \in \mathcal{I}$. For some ancestor set $\mathcal{A}_I$, the alternative distribution is in the form $P(\mathbf{Y} \cup \mathcal{A}_I|\text{do}(Pa(\mathbf{Y} \cup \mathcal{A}_I)), \text{do}(I))$ which should be equivalent to $P(\mathbf{Y} \cup \mathcal{A}|Pa(\mathbf{Y} \cup \mathcal{A}), \text{do}(I))$. We will find an $\mathcal{A}_I$ from the $D^I, \forall I \in \mathcal{I}$ such that we do not require $Pa(\mathbf{Y} \cup \mathcal{A})$ to be intervened on.

Now, to match $P(\mathbf{Y} \cup \mathcal{A}|Pa(\mathbf{Y} \cup \mathcal{A}), \text{do}(I)) = Q(\mathbf{Y} \cup \mathcal{A}|\text{do}(Pa(\mathbf{Y} \cup \mathcal{A})), \text{do}(I))$ with our generative models, we pick the observations of $Pa(\mathbf{Y} \cup \mathcal{A})$ from D^I dataset and intervene in our DCM with those values besides intervening on $\mathbb{G}_I$. Since we do not need generated samples for $Pa(\mathbf{Y} \cup \mathcal{A})$ from DCM, rather their observations from the given D^I dataset, we do not require them to be trained beforehand. However, the order in which we train c-components matters and we follow the partial order found for $P(\mathcal{V})$ dataset even thought we train with multiple datasets.

For example, in Figure 2a, we have two graphs G and $G_{\overline{Z_1}}$. We follow G's training order for both graphs to train the c-components, i.e., $[\mathbb{G}_{Z_1}, \mathbb{G}_{Z_2}, \mathbb{G}_{Z_3}] \rightarrow [\mathbb{G}_{X_1}, \mathbb{G}_{X_2}]$. Here, for the c-component $\mathbf{Y} = \{Z_1, Z_2, Z_3\}$, we match $P(\mathcal{V})$ c-factor $P_{x_1}(z_1, z_2, z_3)$ and $P_{z_1}(\mathcal{V})$ c-factor $P_{x_1,z_1}(z_2, z_3)$ thus have to find alternative distribution for them. We find the smallest ancestor set $\mathcal{A}_{\emptyset}, \mathcal{A}_{Z_1}$ for these c-factors in both $D^{\emptyset}$ and D^{Z_1} datasets. $\mathcal{A}_{\emptyset} = \emptyset$ satisfies modularity condition for $P(\mathcal{V})$ c-factor and their $Pa(\mathbf{Y} \cup \mathcal{A}) = \{X_1\}$. $\mathcal{A}_{Z_1} = \emptyset$ satisfies modularity condition for $P_{z_1}(\mathcal{V})$ c-factor and their $Pa(\mathbf{Y} \cup \mathcal{A}) = \emptyset$. At step (2/3) in Figure 2a, 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 both $P(z_1, z_2, z_3|x_1) = Q_{x_1}(z_1, z_2, z_3)$ and $P_{x_1,z_1}(z_2, z_3) = Q_{x_1,z_1}(z_2, z_3)$.

Algorithm 3 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)
      $\mathbf{v}_x^f = \text{RunGAN}(\mathbb{G}, \mathbf{x}^r, V_r, \theta_{H_*})$
      $\hat{\mathbf{v}}_x = \epsilon \mathbf{v}_x^r + (1-\epsilon)\mathbf{v}_x^f$
      $L_i^{(t)} = \mathbb{D}_{w_i}(\mathbf{v}_x^f) - \mathbb{D}_{w_i}(\mathbf{v}_x^r) * \lambda(\|\nabla_{\hat{\mathbf{v}}_x} \mathbb{D}_{w_i}(\hat{\mathbf{v}}_x)\|_2 - 1)^2$
    $w_i = Adam(\nabla_{w_i} \frac{1}{m} \sum_{t=1}^{m} L_i^{(t)}, w_i)$
    $G_{loss} = G_{loss} + \frac{1}{m} \sum_{j=1}^{m} -\mathbb{D}_{w_i}(\mathbf{v}_x^f)$
  for $\theta \in \theta_{H_*}$ {All hnode mechanisms} **do**
    $\theta = Adam(\nabla_\theta G_{loss}, \theta)$
Return: $\theta_1, \ldots \theta_n$

Algorithm 4 IsRule2($Y, X, I = \emptyset$ (by default))

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

Now, we train mechanisms of the next c-component $[\mathbb{G}_{X_1}, \mathbb{G}_{X_2}]$ in our training order (step 3/3). We have to match $P(\mathcal{V})$ c-factor $P_{z_1}(x_1, x_2)$ and $P_{Z_1}(\mathcal{V})$ c-factor $P_{z_1}(x_1, x_2)$. Ancestor set $\mathcal{A}_\emptyset = \{Z_1, Z_3\}$ satisfies the modularity condition for $\mathbf{Y} = \{X_1, X_2\}$ with $P(\mathcal{V})$ dataset but $\mathcal{A}_{Z_1} = \emptyset$ a smaller set, satisfies the modularity condition for same c-factor with $P_{z_1}(\mathcal{V})$ dataset. Also, $P_{z_1}(\mathcal{V})$ c-factor is $P_{z_1}(x_1, x_2)$. Thus if we train $[\mathbb{G}_{X_1}, \mathbb{G}_{X_2}]$ with only $P_{Z_1}(\mathcal{V})$ dataset, it will learn both c-factors and converge with $P_{z_1}(x_1, x_2) = Q_{z_1}(x_1, x_2)$. Since we have matched all the c-factors, our DCM will match both $P(\mathcal{V})$ and $P_{Z_1}(\mathcal{V})$ distributions. During training of $[\mathbb{G}_{X_1}, \mathbb{G}_{X_2}]$, we had $\mathcal{A} = \emptyset$ for both observation and interventional c-factors. Therefore, we do not need any pre-trained mechanisms, rather we can directly use the observations from $P_{Z_1}(\mathcal{V})$ dataset as parent values. We define $\mathcal{H}^I$-graph for each $I \in \mathcal{I}$ as below:

Definition D.2 ($\mathcal{H}^I$-graph)**.** For a post-interventional graph $G_{\overline{I}}$, let the set of c-components in $G_{\overline{I}}$ be $\mathcal{C} = \{C_1, \ldots C_t\}$. Choose a partition $\{H_k^I\}_k$ of $\mathcal{C}$ such that the $\mathcal{H}^I$-graph $\mathcal{H}^I = (V_{\mathcal{H}^I}, E_{\mathcal{H}^I})$, defined as follows, is acyclic: $V_{\mathcal{H}^I} = \{H_k^I\}_k$ and for any s, t, $H_s^I \to H_t^I \in E_{\mathcal{H}^I}$, iff $P(H_t^I|\text{do}(pa(H_t^I) \cap H_s^I)) \neq P(H_t^I|pa(H_t^I) \cap H_s^I)$, i.e., do-calculus rule-2 does not hold. Note that one can always choose a partition of $\mathcal{C}$ to ensure $\mathcal{H}^I$ is acylic: The $\mathcal{H}^I$ graph with a single node $H_1^I = \mathcal{C}$ in $G_{\overline{I}}$. Even though $\mathcal{H}^I$ for different I might have different partial order, during training, every $\mathcal{H}^I$ follows the partial order of $\mathcal{H}^\emptyset$. Since its partial order is valid for other H-graphs as well (Proposition D.14).

Here, $\mathcal{H}^I$ is the $\mathcal{H}$-graph constructed from $G_{\overline{I}}$, for $I \in \mathcal{I}$ where $\mathcal{I}$ is the intervention set. $\mathcal{H}^\emptyset$ is the $\mathcal{H}$-graph constructed from G for observational training. $H_k^I :=$ k-th h-node in the $\mathcal{H}^I$ graph. During H^I-graphs construction, we resolve cycles by combining c-components on that cycle into a single h-node. Please check example in Figure 16. After merging all such cycles, $H^I, \forall I \in \mathcal{I}$ will become directed acyclic graphs. The partial order of this graph will indicate the training order that we can follow to train all variables in G. For example in Figure 9, two given datasets D_1 and D_2, imply two different graphs G and G_{Z_1} respectively. $[Z_3] \to [Z_1, Z_2] \to [X_1, X_2]$ is a valid training order for $\mathcal{H}_{Z_1}$, we follow the same order as $\mathcal{H}^\emptyset$. We follow : $[Z_1, Z_2, Z_3] \to [X_1, X_2]$.

We run the subroutine **Contruct-$\mathcal{H}^I$-graph()** in Algorithm 5 to build $\mathcal{H}$-graphs. We check the edge condition at line 7 and merge cycles at line 7 if any. In Figure 2a step (1/3), we build the $\mathcal{H}$-graph $H_1 : [Z_1, Z_2, Z_3] \to H_2 : [X_1, X_2]$ for G.

D.2.2. Training Process of Modular Modular-DCM

We construct the $\mathcal{H}^I$-graph for each $I \in \mathcal{I}$ at Algorithm 6 line 6. Next, we train each h-node $H_k^\emptyset$ of $\mathcal{H}^\emptyset$, according to its partial order $\mathcal{T}$. Since we follow the partial order of $\mathcal{H}^\emptyset$-graph, we remove the superscript to address the hnode. Next, we match alternative distributions for $P_I(\mathcal{V})$ c-factors that correspond to the c-components in H_k. (lines 6-6) We initialize a set

Algorithm 5 Construct-$\mathcal{H}^I$-graph($G, \mathcal{I}$)

Input: Causal Graph G, Intervention set, $\mathcal{I}$
for each $I \in \mathcal{I}$ **do**
  $\mathcal{C} \leftarrow$ get_ccomponents($G_{\overline{I}}$)
  Construct graph $\mathcal{H}^I$ by creating nodes H_j^I as $H_j^I = C_j, \forall C_j \in \mathcal{C}$
for each $I \in \mathcal{I}$ **do**
  for each $H_s^I, H_t^I \in \mathcal{H}^I$ such that $H_s^I \neq H_t^I$ **do**
    if $P(H_t^I|\text{do}(pa(H_t^I) \cap H_s^I)) \neq P(H_t^I|pa(H_t^I) \cap H_s^I)$ **then**
      $\mathcal{H}^I.add(H_s^I \rightarrow H_t^I)$
  $\mathcal{H}^I \leftarrow$ merge($\mathcal{H}^I$, cyc) $\forall cyc \in Cycles(H^I)$
for each $I \in \mathcal{I}$ **do**
  for each $H_j^\emptyset \in \mathcal{H}^\emptyset$ **do**
    $H_j^I = \bigcup_k H_k^I$ such that $\mathcal{V}(H_k^I) \subseteq \mathcal{V}(H_j^\emptyset)$ [All variables in H_k^I h-node is contained in $H_j^\emptyset$ h-node.]
Return: $\{\mathcal{H}^I : I \in \mathcal{I}\}$

Algorithm 6 Modular Training-I($G, \mathcal{I}, \mathbf{D}$)

Input: Causal Graph G, Intervention set $\mathcal{I}$, Dataset $\mathbf{D}$.
Initialize DCM $\mathbb{G}$
$\mathcal{H}^I \leftarrow$ **Construct-H-graphs**($G, \mathcal{I}$)
for each $H_k \in \mathcal{H}^\emptyset$ in partial order **do**
  $\mathcal{A}_\emptyset \leftarrow \mathcal{V}$ //Initialize with all nodes
  for each $S \subseteq An_G(H_k)$ **do**
    if IsRule2($H_k, S, \emptyset$) $= 1$ and $|S| < |\mathcal{A}_\emptyset|$ **then**
      $\mathcal{A}_\emptyset \leftarrow S$
  for each $I \in \mathcal{I} \cap H_k$ **do**
    $\mathcal{A}_I \leftarrow \mathcal{V}$ //Initialize with all nodes
    for each $S \subseteq An_{G_{\overline{I}}}(H_k)$ **do**
      if IsRule2(H_k, S, I) $= 1$ and $|S| < |\mathcal{A}_I|$ **then**
        $\mathcal{A}_I \leftarrow S$
  $\mathbb{G}_{H_k} \leftarrow$TrainModule($\mathbb{G}_{H_k}, G, H_k, \mathcal{A}, \mathbf{D}$)
Return: $\mathbb{G}$

$\mathcal{A}_I = \{V : V \subseteq An_{G_{\overline{I}}}(H_k)\}, \forall I \in \mathcal{I}$ to keep track of the joint distribution we need to match to train each h-node H_k from D^I datasets. We iterate over each intervention and search for the smallest set of ancestors $\mathcal{A}_I$ in $G_{\overline{I}}$ such that $\mathcal{A}_I$ satisfies the modularity condition for H_k in D^I dataset tested by Algorithm 4: IsRule2(.) (line 6)

$\mathcal{A}_I, \forall I \in \mathcal{I}$ implies a set of joint distributions in Equation 9, which is sufficient for training the current h-node H_k to learn the c-factors $P_{Pa(C_i)\cup I}(C_i), \forall C_i \in H_k, \forall I \in \mathcal{I}$.

$$Q(H_k \cup \mathcal{A}_I|\text{do}(pa(H_k \cup \mathcal{A}_I)), \text{do}(I)) = P(H_k \cup \mathcal{A}_I|pa(H_k \cup \mathcal{A}_I), \text{do}(I)), \text{in } G_{\overline{I}}, \forall I \in \mathcal{I}. \\ \text{Training: } H_k, \text{ Pre-trained: } \mathcal{A}_I, \text{ From } D^I \text{ dataset: } pa(H_k \cup \mathcal{A}_I), \text{ Intervened: do}(I) \tag{9}$$

Training H_k with the $\mathcal{A}_\emptyset$ found at this step, is sufficient to learn the c-factors $P_{Pa(C_i)}(C_i), \forall C_i \in H_k$. Similarly, if we have an interventional dataset with $I \in H_k$ i.e., the intervened variable lies in the current h-node, we have to match c-factors $P_{Pa(C_i)\cup I}(C_i), \forall C_i \in H_k$. To find alternatives to these c-factors, we look for the ancestor set $\mathcal{A}_I$ in the D^I dataset. For each $\mathcal{A}_I$, we train H_k to match the interventional joint distribution in Equation 9. We ignore intervention on any descendants of H_k since the intervention will not affect c-factors differently than the c-factor in the $D^\emptyset$ observational dataset.

D.2.3. Learn $P(\mathcal{V})$-factors from Interventional Datasets

When we need the alternative distribution for $P(\mathcal{V})$ c-factor, we search for the smallest ancestor set in $D^\emptyset$ dataset. However, when we have a dataset D^I with $I \in An_G(H_j^\emptyset)$, we can search for an ancestor set in $An_{G_{\overline{I}}}(H_k)$ and train on D^I to match a distribution that would be a proxy to $P(\mathcal{V})$ c-factor. This is possible because when we factorize $P_I(\mathcal{V})$ for $I \in An_G(H_j^\emptyset)$, the c-factors corresponding to the descendant c-components of I are similar to $P(\mathcal{V})$ c-factors of the same c-components.

We update Theorem 4.5 for interventional datasets as below.

Theorem D.3. *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 6: **Modular-DCM Modular Training-I** on observational and interventional datasets $\mathbf{D}^I \sim P_I(\mathcal{V}), \forall I \in \mathcal{I}$ converges for each h-node in the $\mathcal{H}^{\emptyset}$-graph constructed from $G = (\mathcal{V}, \mathcal{E})$ and DCM induces the distribution $Q_I(V), \forall I \in \mathcal{I}$. Then, we have $i) P_I(V) = Q_I(V)$, and $ii)$ for any interventional causal query $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V})$ that is identifiable from $\mathbf{D}^I, \forall I \in \mathcal{I}$, we have $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V}) = \mathcal{K}_{\mathcal{M}2}(\mathcal{V})$.*

We provide the proof in Appendix D.6.

D.3. Training following the $\mathcal{H}$-graph

Modular-DCM utilizes conditional generative models such as diffusion models and Wasserstein GAN with penalized gradients (Gulrajani et al., 2017) for adversarial training on $\mathcal{L}_1$ and $\mathcal{L}_2$ datasets in Algorithm 3. $\mathbb{G}$ is the DCM, a set of generators and $\{\mathbb{D}_{w_i}\}$ are a set of critics for each intervention dataset. Here, for intervention $\text{do}(X = x), X \in \mathbf{I}$, $\mathbb{G}^{(x)}(\mathbf{z}, \mathbf{u})$ are generated samples and $v \sim \mathbb{P}_x^r$ are real $\mathcal{L}_1$ or $\mathcal{L}_2$ samples. We train our models by iterating over all datasets and learn $\mathcal{L}_1$ and $\mathcal{L}_2$ distributions (lines 3-3). We produce fake interventional samples at line 3 by intervening on the corresponding node of our architecture with Algorithm 9 RunGAN(). Each critic $\mathbb{D}_x$ will obtain different losses, $L_{X=x}$ by comparing the generated samples with different true datasets (line 3). Finally, at line 3, we update each variable's model weights located at the current hnode based on the accumulated generated loss over each dataset at line 3. After calling Algorithm 3: **TrainModule()** for each of the h-nodes according to the partial order of $\mathcal{H}^{\emptyset}$-graph, Modular-DCM will find a DCM equivalent to the true SCM that matches all dataset distributions. According to, Theorem C.3, it will be able to produce correct $\mathcal{L}_1, \mathcal{L}_2$ samples identifiable from these input distributions (Appendix C.1).

D.4. Essential Theoretical Statements Required for Distributions Matching by Modular-DCM Modular Training

In this section and the following section, we prove some theoretical statements required for our algorithm. Figure 10, illustrates the statements we have to prove and the route we have to follow.

In proposition D.6, we prove the property of a sub-graph having the same set of c-components although we intervene on their parents outside that sub-graph. We use this proposition in Lemma D.7 to show that we can apply c-component factorization for any sub-graph under appropriate intervention. Therefore, in Corollary D.8, we can show that c-component factorization can be applied for h-nodes of the $\mathcal{H}$-graph as well.

We build the above theoretical ground and utilize the statements in section D.5. We show that c-factorization works for the $\mathcal{H}$-graph and Modular Training on h-nodes matches those c-factors. Thus, Modular-DCM will be able to match i) the observational joint distribution $P(\mathcal{V})$ after training on observational data (Proposition D.12) ii) the observational joint distribution $P(\mathcal{V})$ after training on partial observational data and interventional data (Proposition D.13) and iii) the interventional joint distribution $P_I(\mathcal{V}), \forall I \in \mathcal{I}$ after training on observational data and interventional data. (Proposition D.15). The last proposition requires the proof that for all intervention $I \in \mathcal{I}$, the generated $\mathcal{H}^I$ graphs follows the same partial order.

Modular-DCM modular training can now match $P(\mathcal{V})$ and $P_I(\mathcal{V}), \forall I \in \mathcal{I}$ according to Proposition D.13 and Proposition D.15. We can now apply Theorem C.3 to say that Modular-DCM modular training can sample from identifiable interventional distributions after training on $D \sim P(V)$. Finally, Theorem D.17 for observational case is a direct application of Proposition D.12 and TheoremD.16 while Theorem D.18 for interventional case is a direct application of Proposition D.13, Theorem D.15 and Theorem D.16.

We start with some definitions that would be required during our proofs.

Definition D.4 (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 $P(\mathcal{V})$.

Definition D.5 (Sub-graph, $(G_{\overline{I}})_V$)**.** Let G_V be a sub-graph of G containing nodes in V and all arrows between such nodes. $(G_{\overline{I}})_V$ refers to the sub-graph of $G_{\overline{I}}$ containing nodes in V only, such that variable I is intervened on i.e., all incoming edges to I is cut off.

Proposition D.6. *Let $V \in \mathcal{V}, I \in \mathcal{I}$ be some arbitrary variable sets. The set of c-components formed from a sub-graph $(G_{\overline{I}})_V$ with intervention I is not affected by additional interventions on their parents from outside of the sub-graph. Formally,*

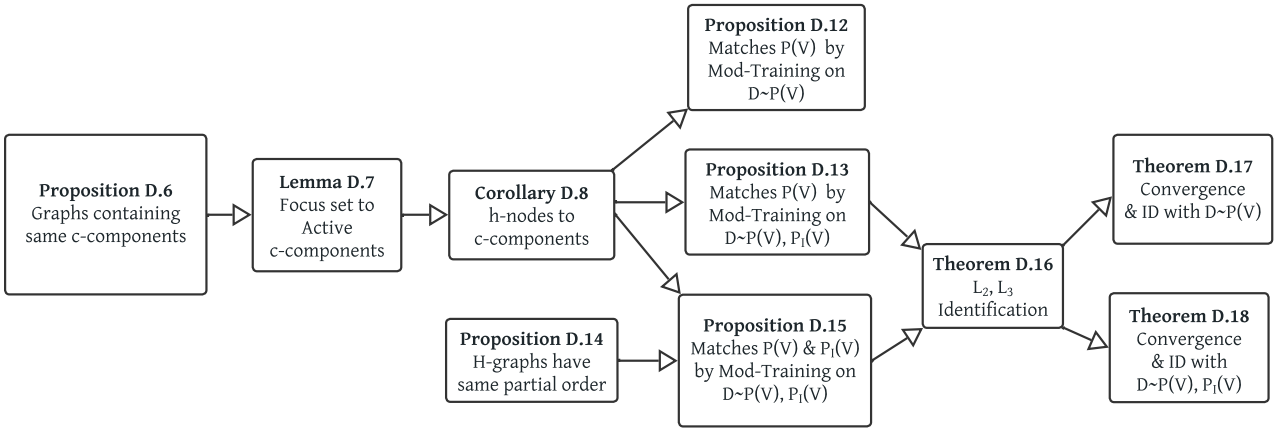


Figure 10. Flowchart of proofs

$(G_{\overline{Pa(V)\cup I}})_V$ and $(G_{\overline{I}})_V$ has the same set of c-components.

Proof. Let $C((G_{\overline{I}})_V)$ be the c-components which consists of nodes of V in graph $(G_{\overline{I}})_V$. In sub-graph $(G_{\overline{Pa(V)\cup I}})_V$, 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 $G_{\overline{I}}$. i.e., $C(G_{V\overline{Pa(V)\cup I}}) = C(G_{\overline{I}})$. □

Lemma D.7. *Let V' be a set called focus-set. V' and intervention I be arbitrary subsets of observable variables $\mathcal{V}$ and $\{C_i\}_i$ be the set of c-components in $G_{\overline{I}}$. Let $\{Pa(V') \cup I\}$ be a set called action-set. and S be a set called remain-set, defined as $S := \mathcal{V}\backslash\{V' \cup Pa(V') \cup I\}$, $S(i)$ 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 $C_i^+ := C_i \setminus \{S(i) \cup Pa(V') \cup I\}$ 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:*

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

Proof Sketch. Similar to the original c-factorization formula $P(\mathcal{V}) = \prod_i P_{Pa(C_i)}(C_i)$, we can factorize as $P_{Pa(V')\cup I}(\mathcal{V}) = \prod_i P_{Pa(C_i)\cup Pa(V')\cup I}(C_i)$. Next, we can marginalize out unnecessary variables S located outside of V' from both sides of this expression. The left hand side of the expression is then $P_{Pa(V')\cup I}(V')$ that is what we need. For the right hand side, we can distribute the marginalization $\sum_S$ among all terms and obtain $\prod_i \sum_{S(i)} P_{Pa(C_i)\cup Pa(V')\cup I}(C_i)$ from $\sum_S \prod_i P_{Pa(C_i)\cup Pa(V')\cup I}(C_i)$. Finally for each product term $P_{Pa(C_i)\cup Pa(V')\cup I}(C_i)$, we remove $S(i)$ from C_i to obtain C_i^+ and drop non parent interventions following do-calculus rule-3. This final right hand side expression becomes, $\prod_i P_{Pa(C_i^+)\cup I}(C_i^+)$. We provide the detailed proof below. □

Proof. $(G_{\overline{Pa(V')\cup I}})_{V'}$ and $(G_{\overline{I}})_{V'}$ have the same c-components according to Proposition D.6. According to Tian's factorization for causal effect identification (Tian & Pearl, 2002), we know that

$$\begin{aligned} P_{Pa(V')\cup I}(\mathcal{V}) &= \prod_i P_{Pa(C_i)\cup Pa(V')\cup I}(C_i) \\ &\quad \text{[let } \eta = Pa(V') \cup I \text{, i.e., action-set]} \\ \implies P_\eta(\eta) \times P_\eta(\mathcal{V} \setminus \eta|\eta) &= \prod_i P_{Pa(C_i)\cup\eta}(C_i) \\ \implies P_\eta(\mathcal{V} \setminus \{Pa(V') \cup I\}) &= \prod_i P_{Pa(C_i)\cup\eta}(C_i) \end{aligned} \tag{10}$$

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

$$
\begin{aligned}
&\implies P_{\eta}(V' \cup S) = \prod_i P_{Pa(C_i)\cup\eta}(C_i) \\
&\text{[Here, } S := \mathcal{V}\backslash\{V'\cup Pa(V') \cup I\} \implies \mathcal{V} \setminus \{Pa(V') \cup I\} = V' \cup S] \\
&\implies \sum_S P_{\eta}(V' \cup S) = \sum_S \prod_i P_{Pa(C_i)\cup\eta}(C_i) \\
&\implies \sum_S P_{\eta}(V' \cup S) = \prod_i \sum_{S(i)} P_{Pa(C_i)\cup\eta}(C_i) \\
&\text{[Since, } S(i) = S \cap C_i \text{and } \forall (i,j), i \neq j, C_i \cap C_j = \emptyset \implies S_i \cap S_j = \emptyset]
\end{aligned} \tag{11}
$$

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 11 helps to visualize all the sets.

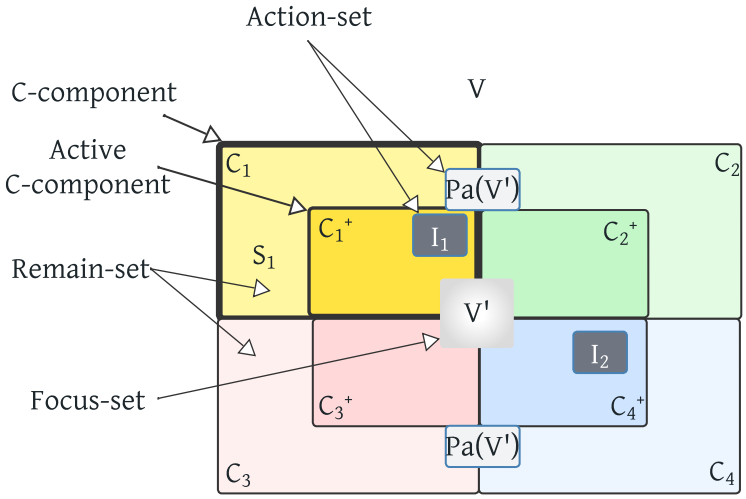


Figure 11. Visualization of focus-sets, action-sets, and remain-sets

We continue the derivation as follows:

$$
\begin{aligned}
&\implies P_{Pa(V')\cup I}(V') = \prod_i P_{Pa(C_i)\cup Pa(V')\cup I}(C_i^+) \\
&\qquad [Here, C_i^+ = C_i \setminus S(i), \text{i.e., active c-component: focus-set elements located in } C_i] \\
&\qquad = \prod_i P_{Pa(C_i^+)\cup I\cup\{Pa(C_i)\setminus Pa(C_i^+)\}\cup Pa(V')}(C_i^+) \\
&\qquad = \prod_i P_{X\cup Z}(C_i^+) \quad [\text{Let, } X = Pa(C_i^+) \cup I \text{ and } Z = \{Pa(C_i) \cup Pa(V')\} \setminus X]
\end{aligned} \tag{12}
$$

Here, we have variable set C_i^+ in the joint distribution. Now, if we intervene on the parents $Pa(C_i^+)$ and I, rest of the intervention which is outside C_i^+ becomes ineffective. Therefore, we have $X = Pa(C_i^+) \cup I$, 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,

$$\implies P_{Pa(V')\cup I}(V') = \prod_i P_{Pa(C_i^+)\cup I}(C_i^+) \quad \text{[We apply Rule 3 since } C_i \perp\!\!\!\perp Z | X_{G_{\overline{X}}}\text{]} \tag{13}$$

□

Corollary D.8, suggests that Tian's factorization can be applied on the h-nodes of $H^I \in \mathcal{H}$.

Corollary D.8. *Consider a causal graph G. Let $\{C_i\}_{i\in[t]}$ be the c-components of $G_{\overline{I}}$. For some intervention target I, let $H^I = (V_{\mathcal{H}^I}, E_{\mathcal{H}^I})$ be the h-graph constructed by Algorithm 5 where $V_{\mathcal{H}^I} = \{H_k^I\}_k$. Suppose H_k^I is some node in $\mathcal{H}^I$. We have that $H_k^I = \{C_i\}_{i\in T_k^I}$ for some $T_k^I \subseteq [t]$. With slight abuse of notation we use H_k^I interchangeably with the set of nodes that are in H_k^I. Then,*

$$P_{Pa(H_k^I)\cup I}(H_k^I) = \prod_{i\in[t]} P_{Pa(C_i)\cup I}(C_i) \tag{14}$$

Proof. Let, $V' = H_k^I$, $C_i^+ = C_i \setminus \emptyset = C_i$. Then, this corollary is direct application of Lemma D.7. □

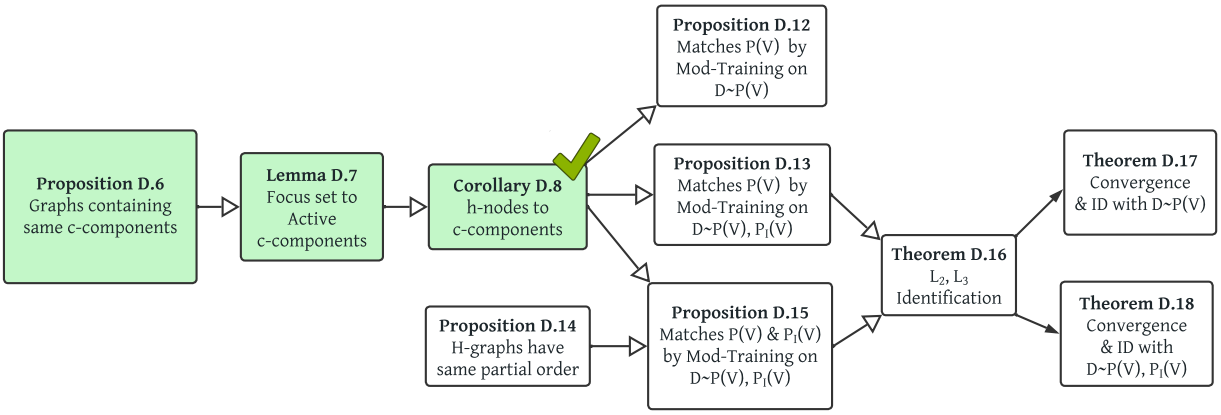


Figure 12. Flowchart of proofs

D.5. Theoretical Proofs of Distributions Matching by Modular-DCM Modular Training

We provide Definition D.2: $\mathcal{H}^I$-graph here again.

Definition D.9 ($\mathcal{H}^I$-graph)**.** For a post-interventional graph $G_{\overline{I}}$, let the set of c-components in $G_{\overline{I}}$ be $\mathcal{C} = \{C_1, \dots C_t\}$. Choose a partition $\{H_k^I\}_k$ of $\mathcal{C}$ such that the $\mathcal{H}^I$-graph $\mathcal{H}^I = (V_{\mathcal{H}^I}, E_{\mathcal{H}^I})$, defined as follows, is acyclic: $V_{\mathcal{H}^I} = \{H_k^I\}_k$ and for any s, t, $H_s^I \to H_t^I \in E_{\mathcal{H}^I}$, iff $P(H_t^I | \text{do}(pa(H_t^I) \cap H_s^I)) \neq P(H_t^I | pa(H_t^I) \cap H_s^I)$, i.e., do-calculus rule-2 does not hold. Note that one can always choose a partition of $\mathcal{C}$ to ensure $\mathcal{H}^I$ is acylic: The $\mathcal{H}^I$ graph with a single node $H_1^I = \mathcal{C}$ in $G_{\overline{I}}$. Even though $\mathcal{H}^I$ for different I might have different partial order, during training, every $\mathcal{H}^I$ follows the partial order of $\mathcal{H}^\emptyset$. Since its partial order is valid for other H-graphs as well.
Training order, $\mathcal{T}$: We define a training order, $\mathcal{T} = \{\sigma_0, \dots, \sigma_m\}$ where $\sigma_i = \{H_k\}_k$. If $H_{k_1}^I \to H_{k_2}^I, H_{k_1}^I \in \sigma_i, H_{k_2}^I \in \sigma_j$ then $i < j$.

Definition D.10 (Notation for distributions)**.** $Q(.)$ is the observational distribution induced by the deep causal SCM. $P(.)$ is the true (observational/interventional) distribution. With a slight abuse of notation, if we have $P(\mathbf{V})$ and intervention I, then $P_I(\mathbf{V})$ indicates $P_I(\mathbf{V} \setminus I)$. Algorithm 6 is said to have converged if training attains zero loss every time line 6 is visited.

Definition D.11 (Ancestor set $\mathcal{A}_I$ in $G_{\overline{I}}$)**.** 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 $H_k^I \in \mathcal{H}^I$-graph, we define $\mathcal{A}_I :=$ the minimal subset of ancestors exists in the causal graph G_I with intervention I such that the following holds,

$$p(H_n^I \cup \mathcal{A}_I | \text{do}(pa(H_n^I \cup \mathcal{A}_I)), \text{do}(I) = p(H_n^I \cup \mathcal{A}_I | pa(H_n^I \cup \mathcal{A}_I), \text{do}(I)) \tag{15}$$

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

$$P(H_k^\emptyset \cup \mathcal{A}_\emptyset | \text{do}(pa(H_k^\emptyset \cup \mathcal{A}_\emptyset)) = P(H_k^\emptyset \cup \mathcal{A}_\emptyset | pa(H_k^\emptyset \cup \mathcal{A}_\emptyset)) \tag{16}$$

Similarly, for $I \in An_G(H_k^I)$, i.e., intervention on ancestors, we can learn $P_{pa(H_k^\emptyset)}(H_k^\emptyset)$ from available interventional datasets since $H_k^I = H_k^\emptyset$, i.e., contains the same c-factors, according to $\mathcal{H}^I$-graphs construction. These c-factor distributions are identifiable from $P_I(V)$ as they can be calculated from the c-factorization of $P_I(V)$. Thus we have,

$$P_{pa(H_k^I)}(H_k^I) = P_{pa(H_k^\emptyset)}(H_k^\emptyset) \tag{17}$$

Therefore, to utilize ancestor interventional datasets, We search for smallest ancestor set $\mathcal{A}_I \subseteq An_{G_{\overline{I}}}(H_k^I)$ in $G_{\overline{I}}$ such that do-calculus rule-2 applies,

$$P(H_k^I \cup \mathcal{A}_I | \text{do}(pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I)) = P(H_k^I \cup \mathcal{A}_I | pa(H_k^I \cup \mathcal{A}_I), \text{do}(I)) \tag{18}$$

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

$$\begin{aligned} & P(H_k^I \cup \mathcal{A}_I | pa(H_k^I \cup \mathcal{A}_I), \text{do}(I)) = Q(H_k^I \cup \mathcal{A}_I | \text{do}(pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I)) \\ \implies & P(H_k^I \cup \mathcal{A}_I | \text{do}(pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I) = Q(H_k^I \cup \mathcal{A}_I | \text{do}(pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I)) \end{aligned} \tag{19}$$

Matching the alternative distributions with D^I will imply that we match $P(\mathcal{V})$ c-factor as well. Formally:

$$\begin{aligned} Q_{pa(H_k^I)}(H_k^I) &= P_{pa(H_k^I)}(H_k^I) \\ Q_{pa(H_k^I)}(H_k^I) &= P_{pa(H_k^\emptyset)}(H_k^\emptyset) \end{aligned} \tag{20}$$

D.5.1. Matching Observational Distributions with Modular Training on $D \sim P(\mathcal{V})$

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

Proposition D.12. *Suppose Algorithm 1:* ***Modular-DCM Modular Training*** *converges for each h-node in $\mathcal{H}^\emptyset$-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,*

$$P(\mathcal{V}) = Q(\mathcal{V}) \tag{21}$$

Proof Sketch. After expressing the observational distribution $P(V)$ as c-factorization expression, we can combine multiple c-factors located in the same h-node as $\prod_{C_i \in H_k} P_{pa(C_i)}(C_i) = P_{pa(H_k)}(H_k)$ according to Corollary D.8. Therefore, if we can match $P_{pa(H_k)}(H_k), \forall k$, this will ensure that we have matched all c-factors $P_{pa(C_i)}(C_i), \forall i$ and as a result $P(V)$ as well. For the h-nodes which does not have any parents (i.e., root nodes) in the $\mathcal{H}$-graph, we know $P_{Pa(H_k)}(H_k) = P(H_k | Pa(H_k))$ due to the construction of H-graph. Therefore, Modular-DCM trains mechanism in those h-nodes by matching $P(H_k | Pa(H_k)) = Q_{Pa(H_k)}(H_k)$.

For the h-nodes which are not root h-nodes in the $\mathcal{H}$-graph, we match $P_{pa(H_k)}(H_k)$ by matching an alternative distribution $P(H_k \cup \mathcal{A} | \text{do}(Pa(H_k \cup \mathcal{A})))$. This alternative distribution factorizes as $P(H_k \cup \mathcal{A} | \text{do}(Pa(H_k \cup \mathcal{A}))) = P_{pa(H_k)}(H_k) *$

$\prod_{H_S \in \mathcal{A}} \prod_{C_j^+ \subseteq H_S} P_{Pa(C_j^+)}(C_j^+)$. Therefore, $P_{Pa(H_k)}(H_k) = \frac{P(H_k \cup \mathcal{A} | \text{do}(Pa(H_k \cup \mathcal{A})))}{\prod_{H_S \in \mathcal{A}} \prod_{C_j^+ \subseteq H_S} P_{Pa(C_j^+)}(C_j^+)}$. The numerator is already matched as that is the alternative distribution we have matched while training h-node H_k. As we are following the topological order of the $\mathcal{H}$-graph, the denominator is matched while training the ancestor h-nodes of H_k. Therefore, Modular-DCM modular training matches $P_{pa(H_k)}(H_k), \forall k$ and thus $P(V) = Q(V)$. We provide the detailed proof below. □

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

$$P(\mathcal{V}) = P(\mathcal{H}) = \prod_{H_k \in \mathcal{H}} \prod_{C_i \in H_k} P_{pa(C_i)}(C_i) \tag{22}$$

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 D.8, we can combine the c-factors in each partitions and rewrite it as:

$$\prod_{H_k \in \mathcal{H}} \prod_{C_i \in H_k} P_{pa(C_i)}(C_i) = P_{pa(H_0)}(H_0) \times P_{pa(H_1)}(H_1) \times \dots \times P_{pa(H_n)}(H_n) \tag{23}$$

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 5, the following is true for any root nodes, $H_k \in \sigma_0$.

$$P(H_k | Pa(H_k)) = P_{Pa(H_k)}(H_k) \tag{24}$$

Modular-DCM training convergence for the DCM in $H_k \in \sigma_0$. (Algorithm 1, line 1) ensures that the following matches:

$$\begin{aligned} P(H_k | Pa(H_k)) &= Q_{Pa(H_k)}(H_k) \\ \implies P_{Pa(H_k)}(H_k) &= Q_{Pa(H_k)}(H_k) \end{aligned} \tag{25}$$

Since, Equation 24 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 D.7 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:

$$\begin{aligned} P(H_k \cup \mathcal{A} | \text{do}(Pa(H_k \cup \mathcal{A}))) &= \prod_{C_i \in H_k} P_{pa(C_i)}(C_i) \times \prod_{H_S \in \{\mathcal{A}\}} \prod_{C_j^+ \subseteq H_S} P_{Pa(C_j^+)}(C_j^+) \\ &\quad \text{[Here, 1st term is the factorization of the current h-node} \\ &\quad \text{and 2nd term is the factorization of the ancestors set.]} \\ \implies P(H_k \cup \mathcal{A} | \text{do}(Pa(H_k \cup \mathcal{A}))) &= P_{pa(H_k)}(H_k) * \prod_{H_S \in \mathcal{A}} \prod_{C_j^+ \subseteq H_S} P_{Pa(C_j^+)}(C_j^+) \end{aligned} \tag{26}$$

Here according to Corollary D.8, 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:

$$\begin{aligned} \implies P_{Pa(H_k)}(H_k) &= \frac{P(H_k \cup \mathcal{A} | \text{do}(Pa(H_k \cup \mathcal{A})))}{\prod_{H_S \in \mathcal{A}} \prod_{C_j^+ \subseteq H_S} P_{Pa(C_j^+)}(C_j^+)} \\ \implies P_{Pa(H_k)}(H_k) &= \frac{Q(H_k \cup \mathcal{A} | \text{do}(Pa(H_k \cup \mathcal{A})))}{\prod_{H_S \in \mathcal{A}} \prod_{C_j^+ \subseteq H_S} Q_{Pa(C_j^+)}(C_j^+)} \end{aligned} \tag{27}$$

Here the R.H.S numerator follows from previous line according to Equation 19. 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 D.7, 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 27 we get,

$$\begin{aligned} & P_{pa(H_k)}(H_k) = Q_{pa(H_k)}(H_k) \\ \implies & P_{pa(H_k)}(H_k) = Q_{pa(H_k)}(H_k) \quad \text{[According to Equation 17]} \end{aligned} \tag{28}$$

Similarly, we train each h-node following the training order $\mathcal{T}$ and match the distribution in Equation 23. This finally shows that,

$$P(V) = \prod_{j \leq n} P_{pa(H_j)}(H_j) = \prod_{j \leq n} Q_{pa(H_j)}(H_j) = Q(V) \tag{29}$$

□

D.5.2. Matching Observational Distributions with Modular Training on $\mathbf{D} \sim P_I(\mathcal{V}), \forall I \in \mathcal{I}$

Now, we provide the theoretical proof of the correctness of Modular-DCM Modular Training matching observational distribution from multiple datasets $D^I, \forall I \in \mathcal{I}$.

Notations: When we consider multiple interventions $I \in \mathcal{I}$, we add I as subscript to each notation to indicate the intervention that notation correspond to. The following notations are mainly used in Proposition D.13, Proposition D.14 and Proposition D.15.

- $\mathbf{D}^I$: the interventional dataset collected with intervention on node I.
- $P_I(V), Q_I(V)$: the interventional joint distribution after intervening on node I representing respectively the real data and the generated data produced from the Modular-DCM DCM.
- $G_{\bar{I}}$ and $\mathcal{H}^I$: the causal graph and the H-graph after do(I) intervention. $\mathcal{H}^\emptyset$ or only $\mathcal{H}$ implies the H-graph for the original causal graph.
- H_k^I: the k-th hnode in the $\mathcal{H}^I$-graph.
- $\mathcal{A}_I$: the ancestor set in $G_{\bar{I}}$-graph, required to construct the alternative distribution for the c-component in consideration. Thus, $\mathcal{A}_\emptyset$ or only $\mathcal{A}$ refers to the observational case.
- σ_0, σ_1: the root hnodes and the non-root hnodes in the H-graph in consideration.

Proposition D.13. *Suppose Algorithm 6: **Modular-DCM Modular Training** converges for each h-node in $\mathcal{H}^\emptyset$-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 $\mathbf{D}^I, \forall I \in \mathcal{I}$. Then,*

$$P(\mathcal{V}) = Q(\mathcal{V}) \tag{30}$$

Proof Sketch. The proof of this Proposition follows the same route as Proposition D.12. In both cases, Modular-DCM matches the observational distribution $P(V)$. The only difference between the two setups is that Modular-DCM has access to multiple interventional datasets in this setup which enables matching observational distribution efficiently by utilizing a smaller ancestor set with the joint. An important fact is that even if we have access to do$(I), \forall I \in \mathcal{I}$ datasets and we construct multiple $\mathcal{H}^I$-graphs, we still follow the topological order of $\mathcal{H}^\emptyset$-graph, i.e, H-graph with no intervention. This is valid according to Proposition D.14 since a topological order of $\mathcal{H}^\emptyset$ works for all $\mathcal{H}^I$-graphs even though $\mathcal{H}^I$ are sparser. Also, any node in H_k^I contains the same set of nodes as in $H_k^\emptyset$ for all k.

We can consider any h-node to be either a root h-node or a non-root h-node. Since for the root h-nodes, the ancestor set is empty, we follow the same approach as the observational case and the proof of correctness follows from Proposition D.12. Now, suppose a h-node is not a root node and intervention I is not located inside it. To match the alternative distribution, instead of searching for the ancestor set in only $H^\emptyset$-graph created for observational data, Modular-DCM looks at all H^I-graphs created based on intervention I and chooses the smallest ancestor set. We assume that a smaller ancestor set will make

it easy to match the corresponding alternative distribution. More precisely, instead of matching $P(H_k \cup \mathcal{A}_\emptyset | \text{do}(Pa(H_k \cup \mathcal{A}_\emptyset)))$ from observational H-graph, Modular-DCM matches $P(H_k \cup \mathcal{A}_I | \text{do}(Pa(H_k \cup \mathcal{A}_I)), \text{do}(I))$ where A_I is smallest across all H^I-graphs. Since intervention I is not located inside the h-node, $P(H_k \cup \mathcal{A}_I | \text{do}(Pa(H_k \cup \mathcal{A}_I)), \text{do}(I)) = P(H_k \cup \mathcal{A}_I | Pa(H_k \cup \mathcal{A}_I), I)$, i.e, the interventional alternative distribution is same as an observational conditional distribution. Thus, we train on the interventional dataset to match $P_{pa(H_k^I)}(H_k^I)$ which is equivalent to $P_{pa(H_k^\emptyset)}(H_k^\emptyset)$ for h-nodes that contain no intervention.

Following the above approach, to match $P_{pa(H_k^\emptyset)}(H_k^\emptyset)$ for all h-nodes will eventually match $P(V)$. The rest follows the same proof as Proposition D.12. We provide the detailed proof below. □

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

$$P(\mathcal{V}) = P(H^\emptyset) = \prod_{H_k^\emptyset \in H^\emptyset} \prod_{C_i \in H_k^\emptyset} P_{pa(C_i)}(C_i) \tag{31}$$

We can divide the set of c-components $\mathcal{C} = \{C_1, \dots C_t\}$ into disjoint partitions or h-nodes as $H_k^\emptyset = \{C_i\}_{i \in T_k}$ for some $T_k \subseteq [t]$. Following Corollary D.8, we can combine the c-factors in each partitions and rewrite it as:

$$\prod_{H_k^\emptyset \in H^\emptyset} \prod_{C_i \in H_k^\emptyset} P_{pa(C_i)}(C_i) = P_{pa(H_0^\emptyset)}(H_0^\emptyset) \times P_{pa(H_1^\emptyset)}(H_1^\emptyset) \times \dots \times P_{pa(H_n^\emptyset)}(H_n^\emptyset) \tag{32}$$

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

For any root h-nodes $H_k^\emptyset \in \sigma_0$:
Due to the construction of $\mathcal{H}^\emptyset$ graphs in Algorithm 5, the following is true for any root nodes, $H_k^\emptyset \in \sigma_0$.

$$P(H_k^\emptyset | Pa(H_k^\emptyset)) = P_{Pa(H_k^\emptyset)}(H_k^\emptyset) \tag{33}$$

Modular-DCM training convergence for the DCM in $H_k^\emptyset \in \sigma_0$. (Algorithm 6, line 6) ensures that the following matches:

$$\begin{aligned} P(H_k^\emptyset | Pa(H_k^\emptyset)) &= Q_{Pa(H_k^\emptyset)}(H_k^\emptyset) \\ \implies P_{Pa(H_k^\emptyset)}(H_k^\emptyset) &= Q_{Pa(H_k^\emptyset)}(H_k^\emptyset) \end{aligned} \tag{34}$$

Since, Equation 33 is true, observational data is sufficient for training the mechanisms in $H_k^\emptyset \in \sigma_0$. Thus, we do not need to train on interventional data.

For the h-node $H_k^\emptyset \in \sigma_1$:
Now we show that we can train mechanisms in $H_k^\emptyset$ by matching $P(\mathcal{V})$ c-factors with either $\mathcal{L}_1$ or $\mathcal{L}_2$ datasets. Let us assume, $\exists \mathcal{A}_I \subseteq \sigma_0$ such that $\mathcal{A}_I = An_{G_{\overline{I}}}(H_k^I)$,i.e., ancestors set of H_k^I in the $\mathcal{H}^I$-graph that we have already trained with available D^I dataset. To apply Lemma D.7 in $G_{\overline{I}}$ with $|I| \geq 0$, consider $V' = H_k^I \cup \mathcal{A}_I$ as the focus-set, $\{Pa(V') \cup I\}$ as the action-set. Thus, active c-components: $C_j^+ := C_j \cap V'$

Then we get the following:

$$\begin{aligned} P(H_k^I \cup \mathcal{A}_I | \text{do}(Pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I)) &= \prod_{C_i \in H_k^I} P_{pa(C_i)}(C_i) \times \prod_{H_S^I \in \{\mathcal{A}_I\}} \prod_{C_j^+ \subseteq H_S^I} P_{Pa(C_j^+) \cup I}(C_j^+) \\ &\quad \text{[Here, 1st term is the factorization of the current h-node} \\ &\quad \text{and 2nd term is the factorization of the ancestors set.]} \\ \implies P(H_k^I \cup \mathcal{A}_I | \text{do}(Pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I)) &= P_{pa(H_k^I)}(H_k^I) * \prod_{H_S^I \in \mathcal{A}_I} \prod_{C_j^+ \subseteq H_S^I} P_{Pa(C_j^+) \cup I}(C_j^+) \end{aligned} \tag{35}$$

Here according to Corollary D.8, we combine the c-factors $P_{pa(C_i)}(C_i)$ for c-components in H_k^I to form $P_{pa(H_k^I)}(H_k^I)$. We continue the derivation as follows:

$$\begin{aligned}
&\implies P_{Pa(H_k^I)}(H_k^I) = \frac{P(H_k^I \cup \mathcal{A}_I | \text{do}(Pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I))}{\prod_{H_S^I \in \mathcal{A}_I} \prod_{C_j^+ \subseteq H_S^I} P_{Pa(C_j^+) \cup I}(C_j^+)} \\
&\implies P_{Pa(H_k^I)}(H_k^I) = \frac{Q(H_k^I \cup \mathcal{A}_I | \text{do}(Pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I))}{\prod_{H_S^I \in \mathcal{A}_I} \prod_{C_j^+ \subseteq H_S^I} Q_{Pa(C_j^+) \cup I}(C_j^+)}
\end{aligned} \tag{36}$$

Here the R.H.S numerator follows from previous line according to Equation 19. For the denominator at R.H.S, the intervention is an ancestor of the current hnode, i.e., $I \in \{An(H_k^I) \setminus H_k^I\}$. Now, $\forall H_S^I \in \mathcal{A}_I$, we have already matched $P(H_S^I \cup \mathcal{A}_I | \text{do}(pa(H_S^I \cup \mathcal{A}_I)), \text{do}(I))$, during training of $\mathcal{A}_I = An(H_k^I)$ h-nodes. According to Lemma D.7, 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 36 we get,

$$\begin{aligned}
&P_{pa(H_k^I)}(H_k^I) = Q_{pa(H_k^I)}(H_k^I) \\
\implies &P_{pa(H_k^\emptyset)}(H_k^\emptyset) = Q_{pa(H_k^\emptyset)}(H_k^\emptyset) \quad \text{[According to Equation 17]}
\end{aligned} \tag{37}$$

Similarly, we train each h-node following the training order $\mathcal{T}$ and match the distribution in Equation 32. This finally shows that,

$$P(V) = \prod_{j \leq n} P_{pa(H_j^\emptyset)}(H_j^\emptyset) = \prod_{j \leq n} Q_{pa(H_j^\emptyset)}(H_j^\emptyset) = Q(V) \tag{38}$$

□

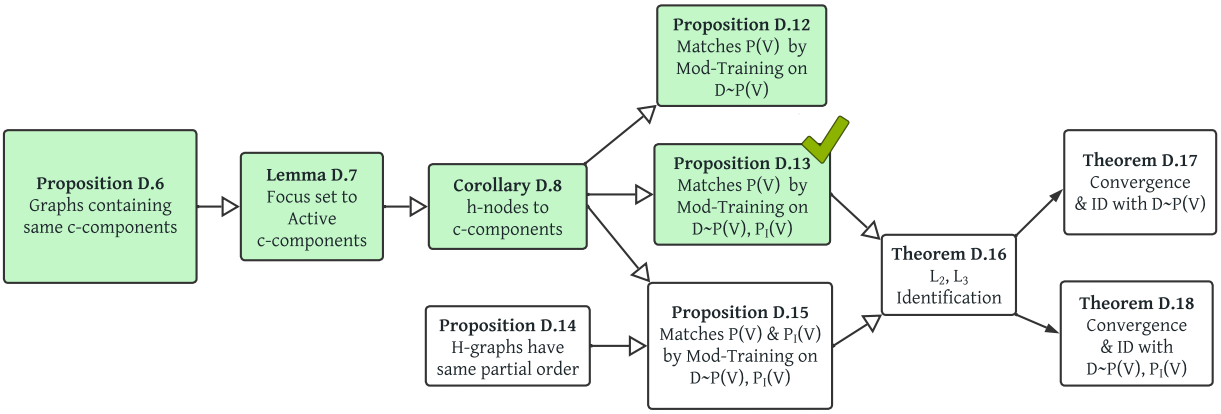


Figure 13. Flowchart of proofs

D.5.3. Matching Interventional Distributions with Modular Training on $\mathbf{D} \sim P_I(\mathcal{V}), \forall I \in \mathcal{I}$

Before showing our algorithm correctness with both observational and interventional data, we first discuss the DAG property of $\mathcal{H}$-graphs. Please check the **notations** in the previous section defined for multiple interventions.

Proposition D.14. *Any $\mathcal{H}$-graph constructed according to Definition D.2 is a directed acyclic graph (DAG) and a common partial order $\mathcal{T}$, exists for all $\mathcal{H}^I$-graphs, $\forall I \in \mathcal{I}$.*

Proof. We construct the $\mathcal{H}$-graphs following Algorithm 5. By checking the modularity condition we add edges between any two h-nodes. However, if we find a cycle $H_j^I, H_k^I \to \ldots \to H_j^I$, then we combine all h-nodes in the cycle and form a

new h-node in $\mathcal{H}^I$. This new h-node contains the union of all outgoing edges to other h-nodes. Therefore, at the end of the algorithm, the final $\mathcal{H}$-graph, $\mathcal{H}^I$ will always be a directed acyclic graph. Note that one can always choose a partition of the c-components $\mathcal{C}$ to ensure $\mathcal{H}^I$ is acylic: The $\mathcal{H}^I$ graph with a single node $H_1^I = \mathcal{C}$.

Next in Algorithm 6, training is performed according to the partial order of $\mathcal{H}^\emptyset$ which corresponds to the original graph G without any intervention. This is the most dense $\mathcal{H}$-graph and thus imposes the most restrictions in terms of the training order. Let I be an intervention set. For any intervention I, suppose $\mathcal{H}^I$-graph is obtained from $G_{\overline{I}}$ and I is located in $H_k^\emptyset$ h-node of $\mathcal{H}^\emptyset$-graph obtained from G. The only difference between $\mathcal{H}^\emptyset$ and $\mathcal{H}^I$ is that the h-node $H_k^\emptyset$ might be split into multiple new h-nodes in $\mathcal{H}^I$ and some edges with other h-nodes that were present in $\mathcal{H}^\emptyset$, might be removed in $\mathcal{H}^I$.

However, according to Algorithm 6, we do not split these new h-nodes rather bind them together to form H_k^I that contain the same nodes as $H_k^\emptyset$. Therefore, no new edge is being added among other h-nodes. This implies that the partial order of $\mathcal{H}^\emptyset$ is also valid for $\mathcal{H}^I$. After intervention no new edges are added to the constructed $\mathcal{H}$-graphs, thus we can safely claim that,

$$An_{G_{\overline{I}}}(H_k^I) \subseteq An_G(H_k^\emptyset), \forall I \in \mathcal{I} \tag{39}$$

Since all $\mathcal{H}$-graphs are DAGs and the above condition holds, any valid partial order for $\mathcal{H}^\emptyset$ is also a valid partial order for all $\mathcal{H}^I, \forall I \in \mathcal{I}$, i.e., they have a common valid partial order.

□

In general, for $I \in H_k^I$, i.e., when the intervention is inside H_k^I, we utilize interventional datasets and search for minimum size variable set $\mathcal{A}_I \subseteq An_{G_{\overline{I}}}(H_k^I)$ in $G_{\overline{I}}$ such that do-calculus rule-2 satisfies,

$$P(H_k^I \cup \mathcal{A}_I | pa(H_k^I \cup \mathcal{A}_I), \text{do}(I)) = P(H_k^I \cup \mathcal{A}_I | \text{do}(pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I)) \tag{40}$$

Then we can train the mechanisms in H_k^I to match the following distribution,

$$\begin{aligned} & P(H_k^I \cup \mathcal{A}_I | pa(H_k^I \cup \mathcal{A}_I), \text{do}(I)) = Q(H_k^I \cup \mathcal{A}_I | \text{do}(pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I)) \\ \implies & P(H_k^I \cup \mathcal{A}_I | \text{do}(pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I) = Q(H_k^I \cup \mathcal{A}_I | \text{do}(pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I)) \end{aligned} \tag{41}$$

Proposition D.15. *Suppose Algorithm 6:* ***Modular-DCM Modular Training*** *converges for each h-node in $\mathcal{H}^\emptyset$-graph constructed from $G = (\mathcal{V}, \mathcal{E})$. Suppose the interventional distribution induced by the deep causal model is $Q_I(V)$ after training on data sets $\mathbf{D}^I, \forall I \in \mathcal{I}$. then,*

$$P_I(V) = Q_I(V) \tag{42}$$

Proof Sketch. The proof of this Proposition follows the same route as Proposition D.12. However, we have now access to both observational and interventional datasets and Modular-DCM is trained on all these datasets modularly to match every interventional joint distribution. An important fact is that even if we have access to $\text{do}(I), \forall I \in \mathcal{I}$ datasets and we construct multiple $\mathcal{H}^I$-graphs, we still follow the topological order of $\mathcal{H}^\emptyset$-graph, i.e, H-graph with no intervention. This is valid according to Proposition D.14 since a topological order of $\mathcal{H}^\emptyset$ works for all $\mathcal{H}^I$-graphs even though $\mathcal{H}^I$ are sparser. Also, any node in H_k^I contains the same set of nodes as in $H_k^\emptyset$ for all k.

Tian's factorization allows us to express the interventional joint distribution $P_I(V)$ in terms of multiple c-factors. We divide the c-components corresponding to these c-factors into two sets. Set-1: the c-component containing the intervention and the c-components in the same h-node. Set-2: the rest of the c-components without any intervention. We combine the c-factors in both sets as H_k^I and $H_{k'}^I \in \{\mathcal{H}^I \setminus H_k^I\}$. Therefore, according to Corollary D.8, $P_I(V)$ can be written as: $P_{pa(C_i)\cup I}(C_i) \times \prod_{H_k^I \in \mathcal{H}^I} \prod_{C_{i'} \in H_k^I} P_{pa(C_{i'})}(C_{i'}) = P_{pa(H_k^I)\cup I}(H_k^I) \times \prod_{H_{k'}^I \in \{\mathcal{H}^I \setminus H_k^I\}} P_{pa(H_{k'}^I)}(H_{k'}^I)$. During the modular training with interventional datasets, Modular-DCM matches each of these c-factors and thus matches the interventional joint distribution.

We can consider any h-node H_k^I as $H_k^I \in \sigma_0$, i.e., to be either a root h-node of $\mathcal{H}^I$ or $H_k^I \in \sigma_1$ i.e., to be a non-root h-node of $\mathcal{H}^I$. For both of these cases, we follow the same approach as the observational case except the fact that we consider h-nodes in the $\mathcal{H}^I$ graph (but the same topological order as $\mathcal{H}^\emptyset$), the ancestor set A_I in $G_{\bar{I}}$ and the $\text{do}(I)$ dataset while matching the interventional distribution for h-nodes. Now, for $H_k^I \in \sigma_0$, by the construction of the $\mathcal{H}^I$ graph, we can say

$P(H_k^I|Pa(H_k^I), \text{do}(I)) = P_{Pa(H_k^I)\cup I}(H_k^I)$. Thus, we match $P_{Pa(H_k^I)\cup I}(H_k^I)$ by training the DCM mechanisms in H_k^I by matching $P(H_k^I|Pa(H_k^I), \text{do}(I)) = Q_{Pa(H_k^I)\cup I}(H_k^I)$.

For h-nodes $H_k^I \in \sigma_1$, we perform modular training to train these mechanisms by matching an alternative interventional joint distribution $P(H_k^I \cup \mathcal{A}_I|\text{do}(Pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I))$ with the do(I) interventional data. This alternative distribution can be expressed as: $P_{pa(H_k^I)\cup I}(H_k^I) \times \prod_{H_S^I \in \mathcal{A}_I} \prod_{C_{i'}^+ \in H_S^I} P_{Pa(C_{i'}^+)}(C_{i'}^+)$. Here the first term correspond to the distribution involving the current h-node H_k^I we are training. The second term corresponds to the partial c-factors located in the ancestors $\mathcal{A}_I$. They are partial C_i^+ because $\mathcal{A}_I$ are ancestors of H_k^I in $G_{\bar{I}}$ satisfying the modularity condition D.1 and not necessarily containing the full c-component C_i. We can equivalently write: $P_{Pa(H_k^I)\cup I}(H_k^I) = \frac{P(H_k^I \cup \mathcal{A}_I|\text{do}(Pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I))}{\prod_{H_S^I \in \mathcal{A}_I} \prod_{C_j^+ \subseteq H_S^I} P_{Pa(C_j^+)}(C_j^+)}$. We match the numerator at the current training step. Since we follow the topological order of the H-graph, the denominator distributions are matched while training the ancestor h-nodes mechanisms in $\mathcal{H}^I$. Therefore, Modular-DCM DCM can match the interventional distribution $P_{Pa(H_k^I)\cup I}(H_k^I)$. More precisely, $Q_{Pa(H_k^I)\cup I}(H_k^I) = P_{Pa(H_k^I)\cup I}(H_k^I)$.

Modular-DCM follows the topological order of $\mathcal{H}^\emptyset$ and trains all mechanisms in any $H_k^\emptyset$. While training the k-th h-node, Modular-DCM enforces the mechanisms in the h-node to learn all interventional distribution $P_{Pa(H_k^I)\cup I}(H_k^I), \forall I \in \mathcal{I}$. Therefore, after training the last node in the topological order, Modular-DCM modular training matches the joint interventional distribution $P_I(V)$. We provide the detailed proof below.

□

Proof. Suppose, intervention I belongs to a specific c-component C_i, i.e., $I \in C_i$. According to Tian's factorization, we can factorize the do(I) interventional joint distributions for $G_{\bar{I}}$ causal graph, into c-factors as follows:

$$P_I(V) = P_I(\mathcal{H}^I) = P_{pa(C_i)\cup I}(C_i) \times \prod_{H_k^I \in \mathcal{H}^I} \prod_{C_{i'} \in H_k^I} P_{pa(C_{i'})}(C_{i'}) \tag{43}$$

The difference between the c-factorization for $P(V)$ and $P_I(V)$ is that when intervention I is located inside c-component C_i, we have $P_{pa(C_i)\cup I}(C_i)$ instead of $P_{pa(C_i)}(C_i)$. We can divide c-components $\mathcal{C} = \{C_1, \dots C_t\}$ into disjoint partitions or h-nodes as $H_k^\emptyset = \{C_i\}_{i \in T_k}$ for some $T_k \subseteq [t]$.

Let, the c-component C_i that contains intervention I belong to hnode H_k^I, i.e., $C_i \in H_k^I$. Following Corollary D.8, we can combine the c-factors in each partitions and rewrite R.H.S of Equation 43 as:

$$P_{pa(C_i)\cup I}(C_i) \times \prod_{H_k^I \in \mathcal{H}^I} \prod_{C_{i'} \in H_k^I} P_{pa(C_{i'})}(C_{i'}) = P_{pa(H_k^I)\cup I}(H_k^I) \times \prod_{H_{k'}^I \in \{\mathcal{H}^I \setminus H_k^I\}} P_{pa(H_{k'}^I)}(H_{k'}^I) \tag{44}$$

Now, we prove that we match each of these terms in Equation 44 according to the training order $\mathcal{T}$.

For any root h-nodes $H_k^I \in \sigma_0$:

Due to the construction of H^I graphs in Algorithm 5, the following is true for any root nodes, $H_k^I \in \sigma_0$.

$$P_I(H_k^I|Pa(H_k^I)) = P_{Pa(H_k^I)\cup I}(H_k^I) \tag{45}$$

Modular-DCM training convergence for the mechanisms in $H_k^I \in \sigma_0$. Algorithm 6, line 6 ensures that the following matches:

$$\begin{aligned} & P_I(H_k^I|Pa(H_k^I)) = Q_{Pa(H_k^I)\cup I}(H_k^I) \\ \implies & P_{Pa(H_k^I)\cup I}(H_k^I) = Q_{Pa(H_k^I)\cup I}(H_k^I) \end{aligned} \tag{46}$$

For the h-node $H_k^I \in \sigma_1$ with $I \in H_k^I$:

Now we show that we can train mechanisms in H_k^I by matching $P_I(\mathcal{V})$ c-factors with $\mathcal{L}_1$ and $\mathcal{L}_2$ datasets. Let us assume, $\exists \mathcal{A}_I \subseteq \sigma_0$ such that $\mathcal{A}_I = An_{G_{\bar{I}}}(H_k^I)$, i.e., ancestors of H_k^I in the $\mathcal{H}^I$-graph that we have already trained with available D^I dataset, $\forall I \in \mathcal{I}$.

To apply Lemma D.7 in $G_{\overline{I}}$ with $|I| \geq 0$, consider $V' = H_k^I \cup \mathcal{A}_I$ as the focus-set, $\{Pa(V') \cup I\}$ as the action-set. Thus, active c-components: $C_j^+ \coloneqq C_j \cap V'$. We apply the lemma as below:

$$P(H_k^I \cup \mathcal{A}_I | \text{do}(Pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I)) = \prod_{C_i \in H_k^I} P_{pa(C_i) \cup I}(C_i) \times \prod_{H_S^I \in \mathcal{A}_I} \prod_{C_{i'}^+ \in H_S^I} P_{Pa(C_{i'}^+)}(C_{i'}^+) \tag{47}$$

Here, the 1st term is the factorization of the current h-node and the 2nd term is the factorization of the ancestors set. The intervened variable I is located in the current h-node H_k^I. Therefore, the factorized c-components, i.e., $C_i \in H_k^I$ has I as intervention along with their parent intervention. The above equation implies:

$$P(H_k^I \cup \mathcal{A}_I | \text{do}(Pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I)) = P_{pa(H_k^I) \cup I}(H_k^I) \times \prod_{H_S^I \in \mathcal{A}_I} \prod_{C_{i'}^+ \in H_S^I} P_{Pa(C_{i'}^+)}(C_{i'}^+) \tag{48}$$

According to Corollary D.8, we combine the c-factors $P_{pa(C_i) \cup I}(C_i)$ for c-components in H_k^I to form $P_{pa(H_k^I) \cup I}(H_k^I)$. We continue the derivation as follows:

$$\begin{aligned} \implies P_{Pa(H_k^I) \cup I}(H_k^I) &= \frac{P(H_k^I \cup \mathcal{A}_I | \text{do}(Pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I))}{\prod_{H_S^I \in \mathcal{A}_I} \prod_{C_j^+ \subseteq H_S^I} P_{Pa(C_j^+)}(C_j^+)} \\ \implies P_{Pa(H_k^I) \cup I}(H_k^I) &= \frac{Q(H_k^I \cup \mathcal{A}_I | \text{do}(Pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I))}{\prod_{H_S^I \in \mathcal{A}_I} \prod_{C_j^+ \subseteq H_S^I} Q_{Pa(C_j^+)}(C_j^+)} \end{aligned} \tag{49}$$

Here the R.H.S numerator follows from previous line according to Equation 41 since training has converged for the current h-node. For the R.H.S, denominator, $\forall H_S^I \in \mathcal{A}_I$ appear before H_k^I in the partial order. When we trained h-nodes $H_S^I \in \mathcal{A}_I$ on $P(\mathcal{V})$ and $P_I(\mathcal{V})$ datasets, we matched the joint distribution $P(H_S^I \cup \mathcal{A}_I | \text{do}(pa(H_S^I \cup \mathcal{A}_I)), \text{do}(I)), \forall H_S^I \in \mathcal{A}_I$. According to Lemma D.7, matching these distribution is sufficient to match the distribution at the R.H.S denominator. Therefore, our DCM will produce the same distribution as well. This implies that from Equation 49 we get,

$$P_{pa(H_k^I) \cup I}(H_k^I) = Q_{pa(H_k^I) \cup I}(H_k^I) \tag{50}$$

Similarly, we train each h-node following the training order $\mathcal{T}$ and match the distribution in Equation 44. We train the c-factor that contains interventions with our available interventional dataset and the c-factors that do not include any interventions can be trained with $P(V)$ dataset. This finally shows that,

$$\begin{aligned} P_I(V) =& P_{pa(H_k^I) \cup I}(H_k^I) \times \prod_{H_{k'}^I \in \{H^I \setminus H_k^I\}} P_{pa(H_{k'}^I)}(H_{k'}^I) \\ =& Q_{pa(H_k^I) \cup I}(H_k^I) \times \prod_{H_{k'}^I \in \{H^I \setminus H_k^I\}} Q_{pa(H_{k'}^I)}(H_{k'}^I) \\ =& Q_I(V) \end{aligned} \tag{51}$$

□

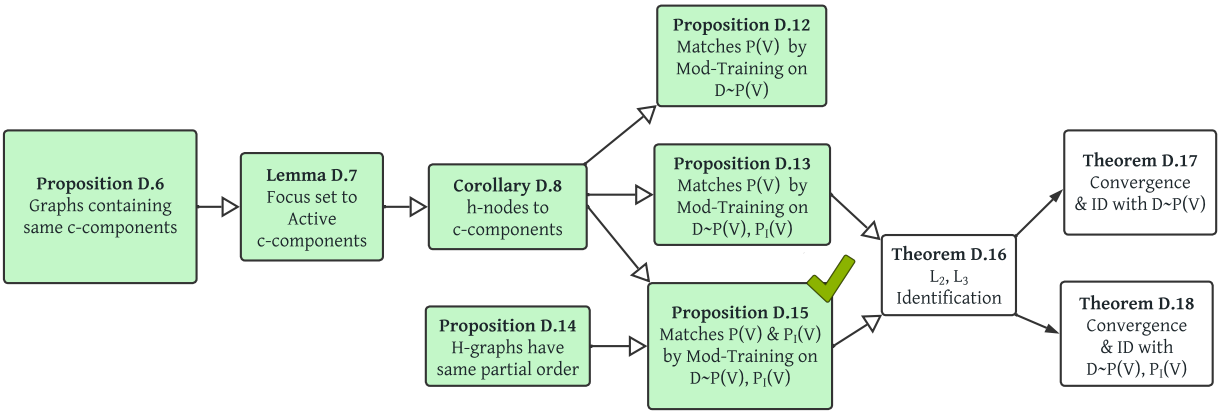


Figure 14. Flowchart of proofs

D.6. Identifiability of Algorithm 6:Modular-DCM Modular Training

Theorem D.16. *Let $\mathcal{M}_1$ be the true SCM and Algorithm 6: **Modular-DCM Modular Training** converge for each h-node in $\mathcal{H}$ constructed from $G = (\mathcal{V}, \mathcal{E})$ after training on data sets $\mathbf{D} = \{\mathcal{D}^I\}_{\forall I \in \mathcal{I}}$ and output the DCM $\mathcal{M}_2$. Then for any $\mathcal{L}_2$ causal query $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V})$, identifiable from $\mathbf{D}$, $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V}) = \mathcal{K}_{\mathcal{M}2}(\mathcal{V})$ holds.*

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 Modular-DCM. For any $H_k^I \in \mathcal{H}^I, I \in \mathcal{I}$, we observe the joint distribution $P(H_k^I \cup \mathcal{A}^I \cup Pa(H_k^I \cup \mathcal{A}^I), \text{do}(I))$ in the input D^I datasets. Thus we can train all the mechanisms in the current h-node H_k^I by matching the following distribution from the partially observable datasets:

$$P(H_k^I \cup \mathcal{A}_I | pa(H_k^I \cup \mathcal{A}_I), \text{do}(I)) = Q(H_k^I \cup \mathcal{A}_I | \text{do}(pa(H_k^I \cup \mathcal{A}_I)), \text{do}(I)) \tag{52}$$

Now, as we are following a valid partial order of the $\mathcal{H}^\emptyset$-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 52. In the same manner, we would be able to match the distributions for each h-node and reach convergence for each of them. Modular-DCM Training convergence implies that $Q_I(\mathcal{V}) = P_I(\mathcal{V}), \forall I \in \mathcal{I}$ i.e., for all input dataset distributions. Therefore, according to Theorem C.3, Modular-DCM is capable of producing samples from correct interventional that are identifiable from the input distributions. □

Theorem D.17. *Suppose Algorithm 1: **Modular-DCM Modular Training** converges for each h-node in the $\mathcal{H}$-graph constructed from $G = (\mathcal{V}, \mathcal{E})$ and after training on observational dataset $D \sim P(\mathcal{V})$, the observational distribution induced by the DCM is $Q(V)$. Then, we have $i) P(V) = Q(V)$, and $ii)$ for any $\mathcal{L}_2$ causal query $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V})$ that is identifiable from $\mathbf{D}$, we have $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V}) = \mathcal{K}_{\mathcal{M}2}(\mathcal{V})$*

Proof. Theorem 4.5 is restated here. The first part of the theorem is proved in Proposition D.12. The second part can be proved with Theorem D.16. □

Theorem D.18. *Suppose Algorithm 6: **Modular-DCM Modular Training** converges for each h-node in the $\mathcal{H}^\emptyset$-graph constructed from $G = (\mathcal{V}, \mathcal{E})$ and after training on observational and interventional datasets $\mathbf{D}^I \sim P_I(\mathcal{V}) \forall I \in \mathcal{I}$, the distribution induced by the DCM is $Q_I(V), \forall I \in \mathcal{I}$. Then, we have $i) P_I(V) = Q_I(V)$, and $ii)$ for any $\mathcal{L}_2$ causal query $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V})$ that is identifiable from $\mathbf{D}^I, \forall I \in \mathcal{I}$, we have $\mathcal{K}_{\mathcal{M}_1}(\mathcal{V}) = \mathcal{K}_{\mathcal{M}2}(\mathcal{V})$*

Proof. The first part of the theorem is proved in Proposition D.13 and Proposition D.15. Then it is a direct implication of Theorem D.16, This theorem is equivalent to Theorem 4.5 if we consider $\mathcal{I} = \{\emptyset\}$. □

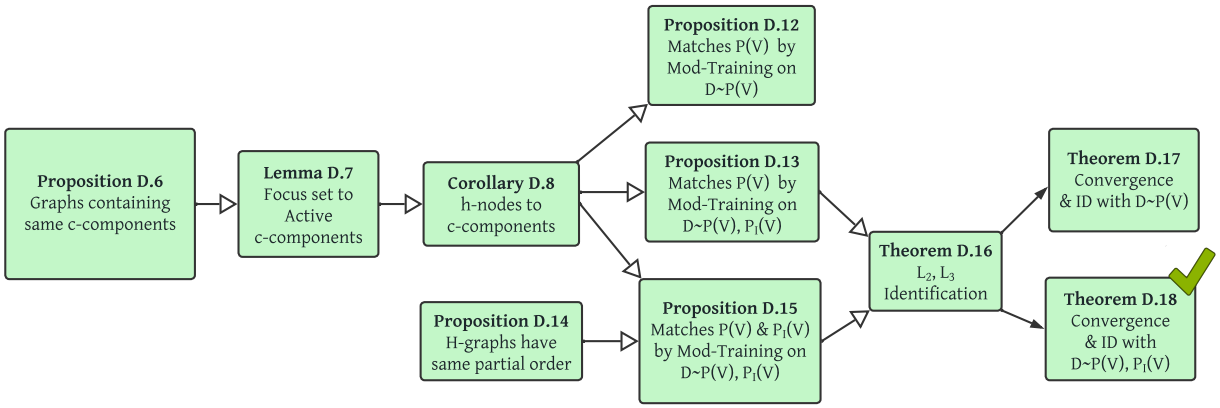


Figure 15. Flowchart of proofs

E. Modular Training on Different Graphs

E.1. Modular Training Example

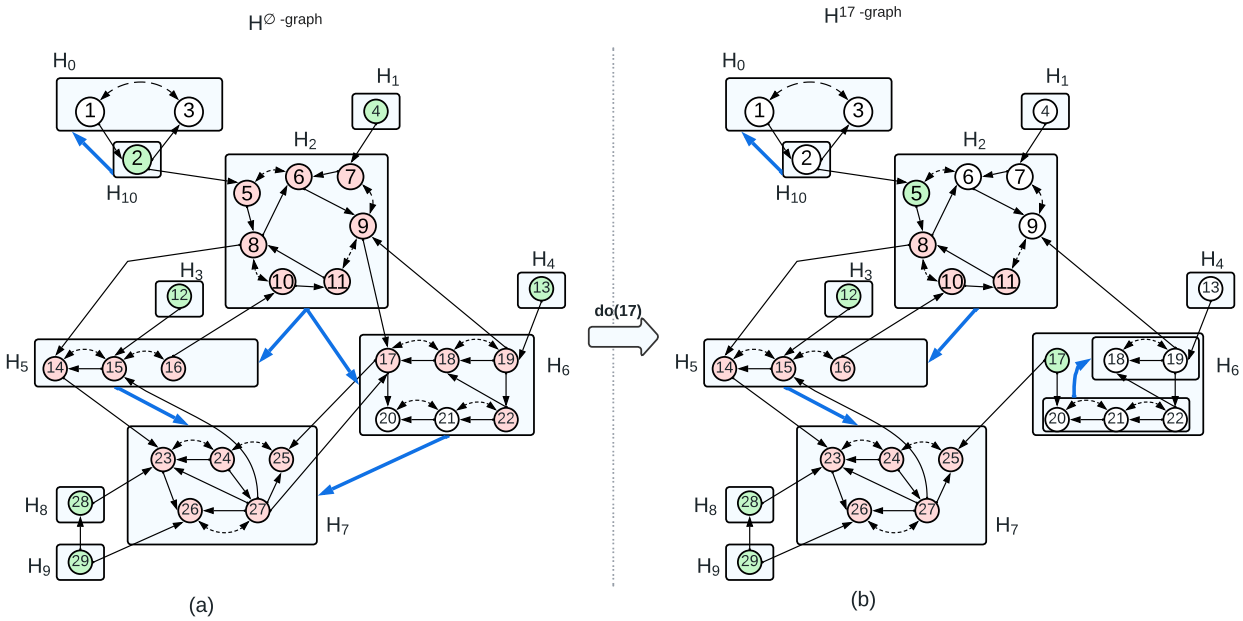


Figure 16. $\mathcal{H}^{\emptyset}$-graph and H^{17}-graph construction

In Figure 16, we construct the $\mathcal{H}^{\emptyset}$-graph as below. We describe the H-graph edges (thick blue edges) and the backdoor path (thin black edges) responsible for those edges.
$H_{10} \to H_0 : 3 \leftarrow 2 \leftarrow 1$
$H_2 \to H_5 : 14 \leftarrow 8 \leftarrow 11 \leftarrow 10 \leftarrow 16,$
$H_2 \to H_6 : 17 \leftarrow 9 \leftarrow 19,$
$H_5 \to H_7 : 23 \leftarrow 14 \leftarrow 15 \leftarrow 27,$
$H_6 \to H_7 : 25 \leftarrow 17 \leftarrow 27$

In Figure 16, we construct the H^{17}-graph as below:
$H_{10} \rightarrow H_0 : 3 \leftarrow 2 \leftarrow 1$
$H_2 \rightarrow H_5 : 14 \leftarrow 8 \leftarrow 11 \leftarrow 10 \leftarrow 16$,
$H_5 \rightarrow H_7 : 23 \leftarrow 14 \leftarrow 15 \leftarrow 27$,
Now, notice that due to $do(17)$, H_6^{17} gets splitted into two new h-nodes, $[18, 19]$ and $[20, 21, 22]$ with a new edge $[20, 21, 22] \rightarrow [18, 19]$. However, according to our H-graph construction algorithm, we keep these two new h-nodes of H^{17} combined inside H_6^{17} same as $\mathcal{H}^{\emptyset}$-graph. Therefore, $\mathcal{H}^{\emptyset}$ and H^{17}'s common partial order does not change.

For training $H_7^{\emptyset}$ node: $\{23, 24, 25, 26, 27\}$, we match the following distribution found by applying do-calculus rule 2.

$$P(23, 24, 25, 26, 27, 14, 15, 16, 17, 18, 19, 22, 5, 6, 7, 8, 9, 10, 11, |\text{do}(2, 12, 13, 28, 29, 4)) \tag{53}$$

In Figure 16(a), joints are shown as red nodes and their parents as green nodes. However, consider, we have both observational and interventional datasets from $P(\mathcal{V})$ and $P(\mathcal{V}|\text{do}(17))$ and we have already trained all the ancestor h-nodes of H_7^{17}. Then we can train the mechanisms that lie in $H_7^{\emptyset}$ to learn both observational and interventional distribution by matching a smaller joint distribution compared to Equation 53:

$$P(23, 24, 25, 26, 27, 8, 10, 11, 14, 15, 16|\text{do}(12, 28, 29, 5, 17)) \tag{54}$$

In Figure 16(b), joints are shown as red nodes and their parents as green nodes. We see that the number of red nodes is less for H^{17} graph compared to $\mathcal{H}^{\emptyset}$ graph when we were matching the mechanisms in h-node, $H_7^{\emptyset}$.

F. Experimental Analysis

In this section, we provide implementation details and algorithm procedures of our Modular-DCM training.

F.1. Training Details and Compute

We performed our experiments on a machine with an RTX-3090 GPU. The experiments took 1-4 hours to complete. We ran each experiment for 300 epochs. We repeated each experiment multiple times to observe the consistent behavior. Our datasets contained $20 - 40K$ samples, and the batch_size was 200, and we used the ADAM optimizer. For evaluation, we generated 20k fake samples after a few epochs and calculated the target distributions from these 20k fake samples and 20k real samples. We calculated TVD and KL distance between the real and the learned 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 it until 0.1. We used different architectures for different experiments since each experiment dealt with different data types: low-dimensional discrete variables and images. Details are provided in the code. For low-dimensional variables, we used two layers with 256 units per layer and with BatchNorm and ReLU between each layer. Please check our code for architectures of other neural networks such as encoders and image generators

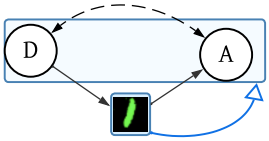
F.2. Complexity Evaluation

Suppose, a causal graph has N variables. Without modularization, we have to match the joint distribution containing N (might be large) number of low and high dimensional variables in a single training phase. Matching that joint distribution with deep-learning models, and a complicated confounded causal structure could be difficult since we are attempting to minimize a very complicated loss function for a very large neural network. Our proposed method allows us to reduce the complexity of this problem tremendously by modularizing the training process to c-components. The size of a c-component is generally a lot smaller than the whole graph. Thus, even though we have to train mechanisms in a c-component together and match a joint distribution involving high and low dimensional variables, the complexity will be much lower. Without our approach, there is no existing work that can modularize and simplify the training process for a causal graph with latents.

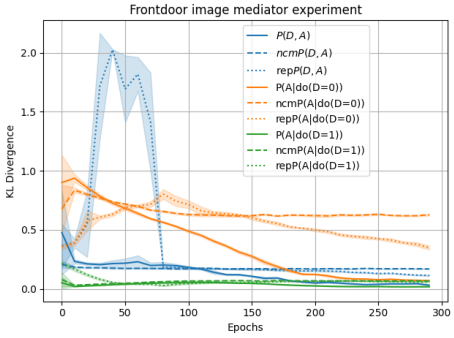
To achieve a deep causal generative model (DCM), given a causal graph of N nodes, it is required to train N neural networks. However, our nearest benchmark NCM, trains all N networks together at the same time. While our method trains only the networks that belong to a single h-node. Thus, during a training phase, the maximum number of networks NCM has to train together is $O(N)$ and in our case, it is $O(|\text{Largest h-node}|)$ which is in most cases $O(|\text{Largest c-component}|)$.

F.3. Image Mediator Experiment

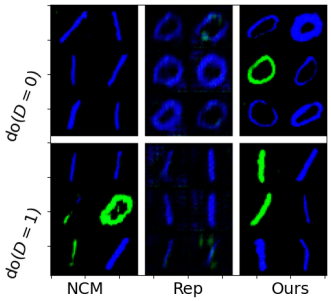
In this section, we provide additional information about the experiment described in Section 5.1. The front-door graph has been instrumental for a long time in the causal inference literature. However, it was not shown before that modular training with high dimensional data was possible, even in the front door graph. This is why we demonstrate the utility of our work on this graph.



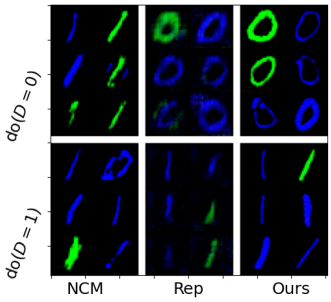
(a) Frontdoor causal graph w/ image mediator



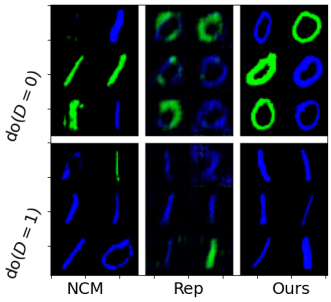
(b) Training converges matching $P(D, A)$ and $P(A|do(D))$



(c) MNIST Image generation comparison



(d) MNIST Image generation comparison



(e) MNIST Image generation comparison

Figure 17. Modular Training on frontdoor causal graph with training order: $\{I\} \rightarrow \{D, A\}$

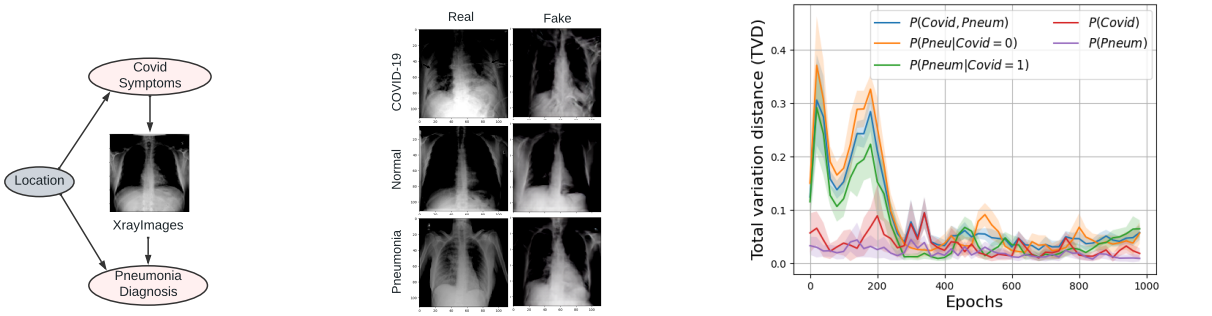
We have domain $D = [0, 1]$, Image size=$3 \times 32 \times 32$ and $C = [0, 1, 2]$. Let U_0, e_1, e_2, e_3 are randomly generate exogenous noise. $D = U_0 + e_1, Image = f_2(D, e_2), A = f_3(Image, e_3, U_0)$. f_2 is a function which takes D and e_2 as input and produces different colored images showing D digit in it. f_3 is a classifier with random weights that takes U_0, e_3 and $Image$ as input and produces A such a way that $|P(A|do(D = 0)) - P(A|D = 0)|, |P(A|do(D = 1)) - P(A|do(D = 0))|and|P(A|D = 1) - P(A|D = 0)|$ is enough distant. The digit color can be considered as exogenous noise. The target is to make sure that the backdoor edge $D \leftrightarrow A$ and the causal path from D to A is active. Since we have access to U_0 as part of the ground truth, we can calculate the true value of $P(A|do(D))$ with the backdoor criterion (Pearl, 1993):

$$P(A|do(D)) = \int_{U_0} P(A|D, U_0)P(D|U_0)$$

During training, U_0 is unobserved but still, the query is identifiable with the front door criterion (Pearl, 2009). Image is a mediator here.

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

However, this inference is not possible with the identification algorithm since it requires image distribution. But Modular-DCM can achieve that by producing Image samples instead of learning the explicit distribution. If we can train all mechanisms in the Modular-DCM DCM to match $P(D, A, I)$, we can produce correct samples from $P(A|do(D))$. We



(a) COVIDx CXR-3 graph (b) Real vs fake (c) Training converges with low TVD

Figure 18. Modular-DCM converges with pre-trained model on COVIDx CXR-3 dataset.

construct the Modular-DCM architecture with a neural network $\mathbb{G}_D$ having fully connected layers to produce D, a deep convolution GAN $\mathbb{G}_I$ to generate images, and a classifier $\mathbb{G}_A$ to classify MNIST images into variable A such that D and A are confounded. Now, for this graph, the corresponding $\mathcal{H}$-graph is $[I] \to [D, A]$. Thus, we first train $\mathbb{G}_I$ by matching $P(I|D)$. Next, to train $\mathbb{G}_D$ and $\mathbb{G}_A$, we should match the joint distribution $P(D, A, I)$ since $\{I\}$ is ancestor set $\mathcal{A}$ for c-component $\{D, A\}$. GAN convergence becomes difficult using the joint distribution loss since the losses generated by low and high dimensional variables are not easily comparable and it is non-trivial to find a correct re-weighting of such different loss terms. To the best of our knowledge, no current causal effect estimation algorithm can address this problem since there is no estimator that does not contain explicit image distribution, which is practically impossible to estimate. To deal with this problem, we map samples of I to a low-dimensional representation, RI with a trained encoder and match $P(D, RI, A)$ instead of $P(D, Image, A)$.

Note that, we use the mechanism training order $[I] \to [D, A]$ specified by the H-graph (Algorithm 7) to match the joint distribution $P(D, Image, A)$. It is not feasible to follow any other sequential training order such as $[D] \to [Image] \to [A]$ as training them sequentially with individual losses can not hold the dependence in $D \leftrightarrow A$. We compare our performance with NCM in Figure 17. We implemented NCM on our architectures as it could not be directly used for images. For estimating the FID scores, we generated 2050 samples from each method and calculated the FID score compared to the original images using a method proposed in (Seitzer, 2020).

F.4. MNIST Diamond Graph

Here we discuss the data generating process of the MNIST diamond graph. We have considered matching the joint distribution for the following diamond graph. $I_1 \to Digit \to I_2 \to Color; I_1 \leftrightarrow Color \leftrightarrow Digit$. Here I_1 and I_2 are image nodes and the rest are discrete. $I_1, Digit, Color$ belong to the same c-component. To generate semi-synthetic data for this graph, we first uniformly sample U_1 and U_2 where $I_1 \leftarrow U_1 \to Digit$ and $Digit \leftarrow U_2 \to Color$. Next, we set $I_1.color$ according to U_1. Then we pick a digit image from the MNIST dataset of $I_1.digit$ and color it with $I_1.color$. Next we generate values for $Digit$ consistent with $I_1.digit$ while adding some confounding variable U_2. We pick another MNIST image with $Digit$ and color it with some random color. Finally we set the value of $Color$ with $I_2.color$ and U_2.

F.5. Performance on Real-world COVIDx CXR-3 Dataset

F.5.1. Real-world COVIDx CXR-3 Dataset

To demonstrate the convergence behavior of Modular-DCM on real high-dimensional datasets, we conduct a case study with the COVIDx CXR-3 (Wang et al., 2020) dataset in this section. This dataset contains 30,000 chest X-ray images ($Xray$) with Covid (C) and pneumonia (N) labels from over 16,600 patients located in 51 countries. Even though there is no ground truth causal graph associated with this dataset, we consider the same motivational setup we discussed in Figure 1a: $C \to Xray \to N, C \leftrightarrow N$. We assume the graph to be consistent with the dataset since it does not impose conditional independence restrictions on the joint distribution $P(C, Xray, N)$. Therefore, we expect our modular training algorithm to correctly match the observational joint distribution. We discuss the reasoning behind each edge in Appendix F.5. We aim to learn that if a patient is randomly picked and intervened with Covid (hypothetically), how likely will they be diagnosed with pneumonia, i.e., $P(N|do(C))$? To match the joint distribution $P(C, Xray, N)$, we follow the modular training order:

$[\mathbb{G}_{Xray}] \to [\mathbb{G}_C, \mathbb{G}_N]$. Instead of training $\mathbb{G}_{Xray}$ from scratch, we use a pre-trained model (Giorgio Carbone, 2023) that can be utilized to produce Xray images corresponding to $C \in [0, 1]$ input. Next, we train $\mathbb{G}_C$ and $\mathbb{G}_N$ together since they belong to the same c-component. Since the joint distribution contains both low and high-dimensional variables, we map Xray to a low-dimensional representation Rxray with an encoder and match $P(C, Rxray, N)$.

Evaluation: Figure 18b shows images for the original dataset (left) and output images (right) from the pre-trained model. In Figure 18c, we plot the total variation distance (TVD) of $P(C), P(N), P(N|C), P(N, C)$. We observe that TVD for all distributions is decreasing. The average treatment effect, i.e., the difference between $E[P(N|do(C = 1))]$ and $E[P(N|do(C = 0))]$ is in $[0.05, 0.08]$ after convergence. This implies that intervention with Covid increases the likelihood of being diagnosed with Pneumonia. However, these results are based on this specific COVIDx CXR-3 dataset and should not be used to make medical inferences without expert opinion.

F.5.2. Detailed discussion on COVIDx CXR-3

In this section, we provide some more results of our experiment on COVIDx CXR-3 dataset (Wang et al., 2020). This dataset contains 30,000 chest X-ray images with Covid (C) and pneumonia (N) labels from over 16,600 patients located in 51 countries. The X-ray images are of healthy patients $(C = 0, N = 0)$, patients with non-Covid pneumonia $(C = 0, N = 1)$, and patients with Covid pneumonia $(C = 1, N = 1)$. X-ray images corresponding to COVID non-pneumonia $(C = 1, N = 0)$ are not present in this dataset as according to health experts those images do not contain enough signal for pneumonia detection. However, to make the GAN training more smooth we replaced a few $(C = 1, N = 1)$ real samples with $(C = 1, N = 0)$ dummy samples. We also normalized the X-ray images before training.

Note that the causal effect estimates obtained via this graph may not reflect the true causal effect since the ground truth graph is unknown and there may be other violations of assumptions such as distribution shift and selection bias. In order to demonstrate the convergence behavior of Modular-DCM on real high-dimensional datasets, we consider the causal graph shown in Figure 18a. However, observe that this graph does not impose conditional independence restrictions on the joint distribution $P(C, Xray, N)$. If our mentioned assumptions (including no selection bias, etc.) are correct, we expect Modular-DCM to correctly sample from interventional distribution after training by Theorem 4.5. Therefore, we expect our modular training algorithm to correctly match the observational joint distribution.

Our reasoning for using this causal graph is as follows: we can assume that Covid symptoms determine the X-ray features and the pneumonia diagnosis is made based on the X-rays. Thus we can add direct edges between these variables. A patient's location is hidden and acts as a confounder because a person's socio-economic and health conditions in a specific location might affect both the likelihood of getting Covid and being properly diagnosed with Pneumonia by local health care. The X-ray images are done by chest radiography imaging examination. Due to the standardization of equipment, we assume the difference in X-ray data across hospital locations is minor and can be ignored. Thus, $\text{Location} \not\to \text{XrayImages}$.

To obtain the low dimensional representation of both real and fake X-ray images, we used a Covid conditional trained encoder. Instead of training $\mathbb{G}_{Xray}$ from scratch, we use a pre-trained model (Giorgio Carbone, 2023) that can be utilized to produce Xray images corresponding to $C \in [0, 1]$ input. Note that, this pre-trained model takes value 0 for Covid, 1 for normal, and 2 for Pneumonia as input and produces the corresponding images. If a fake Covid sample indicates Covid=1, we map it to the 0 input of the pre-trained GAN. If a fake Covid sample indicates Covid=0, this might be either mapped to 1 (normal) or 2 (Pneumonia). Instead of randomly selecting the value, we use the real Pneumonia sample to decide this (either 1 or 2). After that, we produce X-ray images according to the decided input values. Since we are using the GAN-generated fake samples for Covid=1, the computational graph for auto grad is not broken. Rather the mentioned modification can be considered as a re-parameterization trick.

F.6. Invariant Prediction on CelebA-HQ

To reflect the distribution shift in $P(Sex)$, we divide image samples from the CelebA-HQ dataset into train and test domains as the table in Figure 5 and the actual number of samples are given in Table 1. In the test domain, $P(Sex)$ changes while $P(Eyeglass)$ stays fixed.

The prediction of $Eyeglass$ from $Image$ is done by learning the probability distribution $P(Sex|Image)$. Note that, we would like the prediction of $Eyeglass$ to be independent of Sex and $Domain$. This might not work if we learn $P(Eyeglass|Image)$ or $P(Eyeglass|Image, Sex)$ since conditioning will make the prediction depend on $Domain$. Thus, we train `Intervention` classifier on $\mathcal{D}[Eyeglass, Image] \sim P(Eyeglass, Image|\text{do}(Sex))$ following the

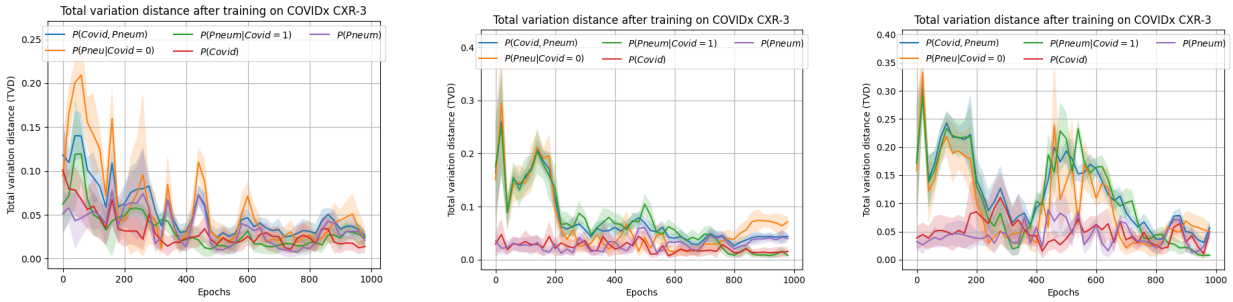


Figure 19. Total variation distance plots show Modular-DCM converges on COVIDx CXR-3 dataset. (consecutive 20 epochs were averaged)

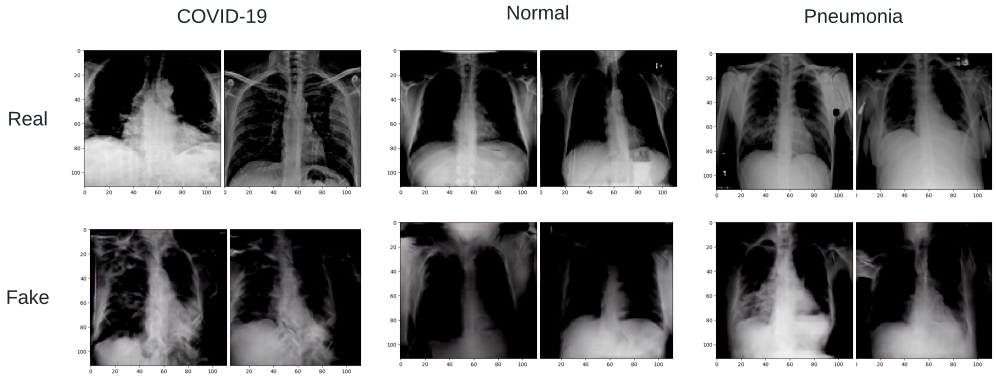


Figure 20. Real images from dataset vs pre-trained GAN generated images

approach suggested in (Subbaswamy et al., 2019).

Since $\{Eyeglass, Sex\}$ and $\{Image\}$ belong to different c-components, we can train models $\mathbb{G}_{Eyeglass}, \mathbb{G}_{Sex}$ together and use a pre-trained model for $\mathbb{G}_I$. Therefore, we only have to train $\mathbb{G}_A$ and $\mathbb{G}_S$. We utilize Modular-DCM's ability to incorporate a pre-trained image generation model, InterFaceGAN (Shen et al., 2020) which can generate impressive human faces in its causal generative models. We generate $10k$ samples of $[Eyeglass', Image'] \sim P(Eyeglass, Image|\text{do}(Sex))$. Intervention on Sex attribute will make $Eyeglass$ independent from both Sex and $Domain$. Finally, the prediction would be independent of the distribution shift in $P(Sex|Domain)$.

We used the InterFaceGAN that uses pre-trained StyleGAN from the repository: `https://github.com/genforce/interfacegan`. To filter incorrect images, we used a pre-trained classifier to from this repository: `https://github.com/clementapa/CelebFaces_Attributes_Classification` to filter the inconsistent images generated from InterFaceGAN.

Table 1. Number of samples in training and test dataset

	Train	
	Eyeglass=0	Eyeglass=1
Sex=0	3200	100
Sex=1	1000	1080
	Test	
Sex=0	400	180
Sex=1	600	100

Figure 21. Image samples generated by InterFaceGAN for $P(Image|Sex, Eyeglass = 1)$

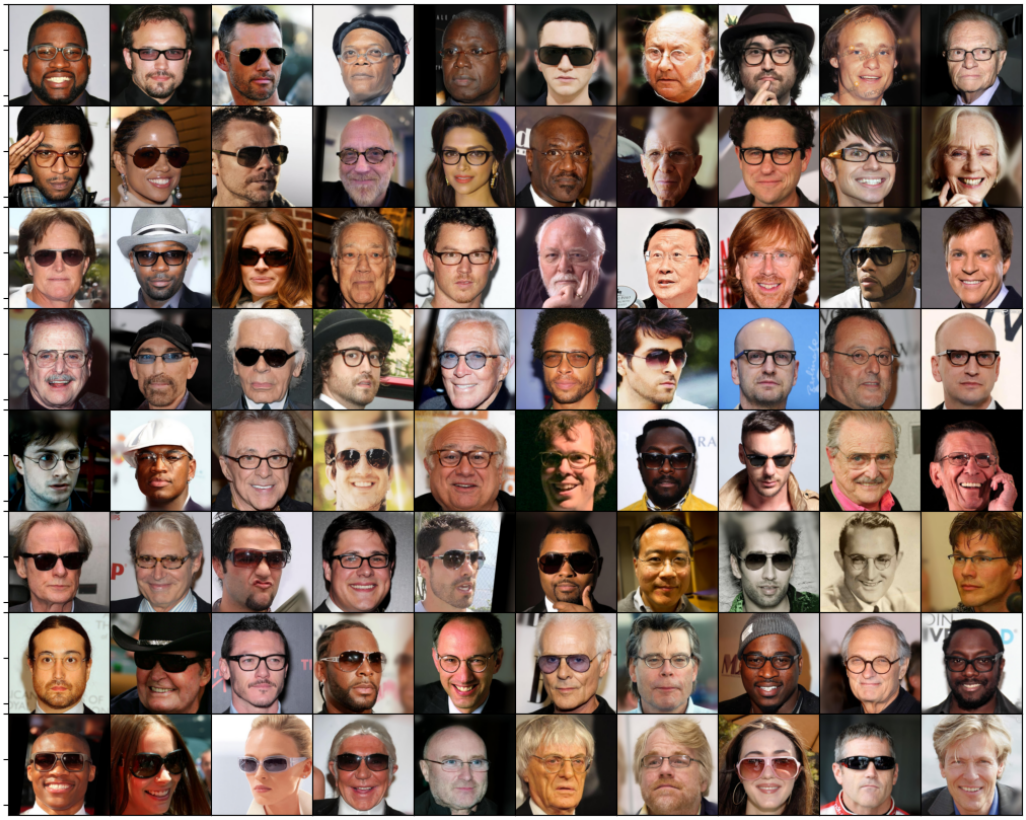
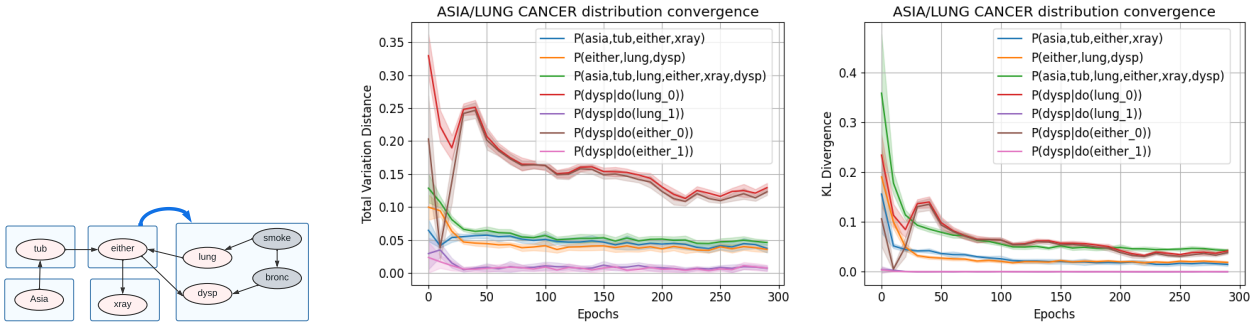


Figure 22. Image samples for $P(Image|Eyeglass = 1)$ from the CelebA-HQ dataset. We observed $P(Sex = 1|Eyeglass = 1)$ is around 0.91 implying that in the training data, there is a high correlation between Sex=Male and wearing eyeglass.

F.7. Asia/Lung Cancer Dataset



(a) Asia causal graph (b) Asia Total Variation Distance (c) KL divergence of Asia experiment

Figure 23. Modular Training on Asia Dataset

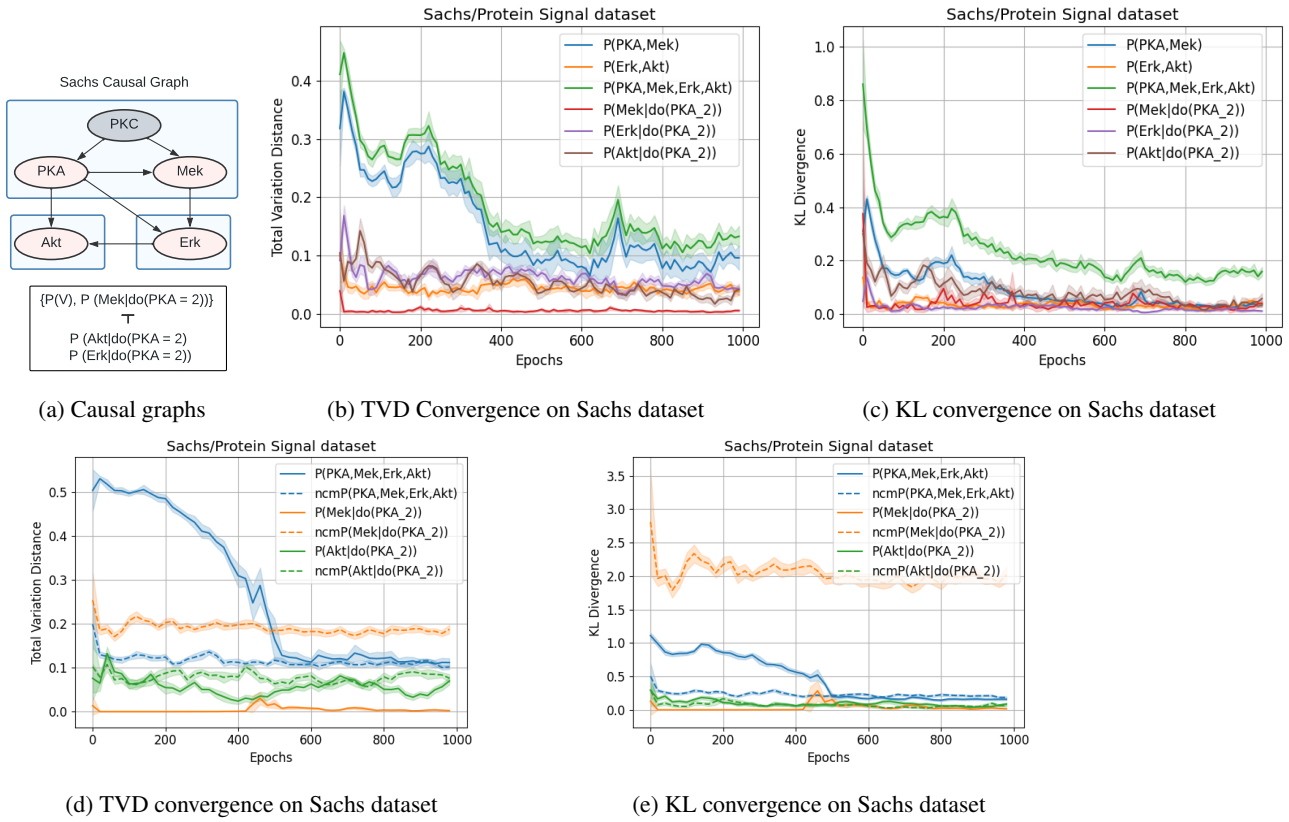
Asia Dataset. We evaluate our algorithm performance on ASIA Dataset from bnlearn repository (Scutari & Denis, 2021). The purpose of this experiment is to show that modular training can learn the joint distribution of the Asia dataset formed as semi-Markovian and correctly produces samples from identifiable $\mathcal{L}_2$ distributions. To check the effectiveness of Modular-DCM for a semi-Markovian causal model, we hide "smoke" and "bronc" variables in the observational dataset as shown in Figure 23a. This action gives us a causal graph with a latent confounder between the "lung" and the "dysp" variables. The $\mathcal{H}$ graph nodes are indicated by the square box containing the variables. According to the algorithm, all $\mathcal{H}$-nodes are disconnected except $[either] \rightarrow [lung, dysp]$. Therefore, we first start training the mechanisms of $asia, tub, either, xray$ and then separately but in parallel train the mechanisms of $lung, dysp$. Here we can also use pre-trained $either$ while we train $lung, dysp$ to match the distribution $P(lung, dysp, either|tub)$. For evaluation, we generated samples from $P(dysp|\text{do}(lung))$ and $P(dysp|\text{do}(either))$ distributions from Modular-DCM. We can calculated $P(dysp|\text{do}(lung))$ with front-door adjustment and $P(dysp|\text{do}(either))$ with back-door adjustment using the real dataset samples. In Figure 23b, 23c, we can see that our partial training is working well with all of the distributions converging to low TVD and KL loss.

F.8. Real-world: Sachs Protein Dataset

For completeness, we test both Modular-DCM and NCM performance on a low-dimensional real-world Sachs dataset (Sachs et al., 2005), which contains a protein signaling causal graph and is given in Figure 24a. The goal is to illustrate Modular-DCM's capability of utilizing multiple partial $\mathcal{L}_1, \mathcal{L}_2$ datasets. We considered the observational dataset $D_1 \sim P(PKA, Mek, Erk, Akt)$ and the interventional dataset $D_2 \sim P(Mek|\text{do}(PKA = 2))$. The interventional dataset with $PKA = 2$ is chosen since it has a large number of samples. Here we intentionally hide variable PKC and considered it as a confounder. Hence, $P(Mek|\text{do}(PKA = 2))$, $P(Akt|\text{do}(PKA = 2)$ and $P(Erk|\text{do}(PKA = 2))$ are non-identifiable from only $P(V)$. According to Corollary C.5, $P(V), P(Mek|\text{do}(PKA = 2))$ make these distributions identifiable. More precisely, if we have access to $P(V)$ and $P_{PKA}(Mek)$ only, then its sufficient to identify, $P_{PKA}(Mek, Erk, Akt)$. We have datasets D_1, D_2 that are sampled from $P(V), P(Mek|\text{do}(PKA = 2))$ for training. If we convert the Sachs graph into the train graph $\mathcal{H}$ in Figure 24a, we see that for only the interventional dataset, we have to train the mechanism of Mek. This is because other variables except Mek belong to different hnodes or training components. Here,

$$\begin{aligned} &P_{PKA}(Mek, Erk, Akt) \\ &= P_{PKA}(Mek)P_{PKA}(Erk|Mek)P_{PKA}(Akt|Erk, Mek) \\ &= P_{PKA}(Mek)P(Erk|Mek, PKA)P(Akt|PKA, Erk, Mek) \end{aligned} \tag{55}$$

Therefore, we train Modular-DCM i.e., the DCM to match both $P(V)$ and $P(Mek|\text{do}(PKA))$. We $i)$ first train $\mathbb{G}_{H_0} = [\mathbb{G}_{PKA}, \mathbb{G}_{Mek}]$ with both D_1 and D_2, $ii)$ next, train $\mathbb{G}_{H_1 \cup H_2} = [\mathbb{G}_{Erk}, \mathbb{G}_{Akt}]$ with only D_1. In Figure 24b and 24c, Modular-DCM converges by training on both $P(V)$ and $P(Mek|\text{do}(PKA = 2))$ datasets. We compared the distributions $P(Akt|\text{do}(PKA = 2)$ and $P(Erk|\text{do}(PKA = 2))$ implicit in Modular-DCM generated samples with the Sachs $\mathcal{L}_2$-dataset distributions and observed them matching with low TVD and KL loss. Even though, we dont observe Erk and Akt in D_2, with modular training, we can still train the mechanisms with $P(V)$ and sample correctly from their $\mathcal{L}_2$- distributions.



(a) Causal graphs (b) TVD Convergence on Sachs dataset (c) KL convergence on Sachs dataset

(d) TVD convergence on Sachs dataset (e) KL convergence on Sachs dataset

Figure 24. Benchmark and Real-world datasets

This reflects the transportability of Modular-DCM. During Modular-DCM training, we can use pre-trained models of $\{PKA, Mek\}, \{Akt\}$ or $\{Erk\}$ since they are located in different hnodes.

***Sachs dataset performance comparison with NCM*:** In Figure 24d and Figure 24e, we compare and show the convergence of both Modular-DCM and NCM with respect to total variation distance and KL-divergence. We observe that for low-dimensional variables, we perform similarly to DCM or better in some cases. However, they do not have the ability to utilize pre-trained models like we do. Besides, unlike NCM, we do not need to run the algorithm again and again for each identifiable queries. Thus, when queries are identifiable, our algorithm can be utilized as an efficient method to train on datasets involving low-dimensional variables.

G. Algorithms & Pseudo-codes

Algorithm 7 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 $P(H_t|\text{do}(pa(H_t) \cap H_s)) \neq P(H_t|pa(H_t) \cap H_s)$ **then**
    $\mathcal{H}.add(H_s \rightarrow H_t)$
$\mathcal{H} \leftarrow$ Merge($\mathcal{H}$, cyc), $\forall cyc \in Cycles(\mathcal{H})$
Return: $\mathcal{H}$

Algorithm 8 isIdentifiable($G, \mathcal{I}, query$)

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

Algorithm 9 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]
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
        $v_i = \mathbb{G}_{\theta_i}(exos, conf, gumbel, \hat{\mathbf{v}}_{par})$
    $\hat{\mathbf{v}} \leftarrow Append(\hat{\mathbf{v}}, v_i)$
Return Samples **v** or Fail

Algorithm 10 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
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($G, \mathbb{G}, [Y], \{X : x\}, \emptyset$)
Return $samples$