Causal Interpretation of Neural Networks

SENHUI GUO

B.Sci

THE UNIVERSITY OF SYDNEY

Supervisor: Wanli Ouyang

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

School of Electrical and Information Engineering
Faculty of Engineering
The University of Sydney
Australia

21 September 2023

Abstract

In recent years, neural networks (NNs) have achieved great success in various challenging tasks. However, due to their black-box nature, it is still challenging to interpret how they process the data. As a result, NNs misled by spurious correlations often fail on out-of-distribution (o.o.d.) samples. Causal theory, on the other hand, has shown great potential in battling spurious correlations and interpreting complex systems. In this work, we use causal theory to tackle two tasks: NN modularity and causal representation learning. First, inspired by structural causal models (SCM), we propose a structural mechanism identification framework, namely, *competitive disentanglement*, to discover modular structures in NNs. The idea is to let neurons compete with each other in a specific setting such that neurons belonging to different modules will exhibit different behaviors. By observing these behaviors of neurons, we are able to disentangle them into different modules. Second, we explore the formulation of causal representation learning. The probability of causation (POC) has been regarded as a good measure of causal relevance. However, due to its counterfactual form, people have not been able to estimate the POC of representations consistently. We propose to model the representation learning process with a disentangled Variational Autoencoder (VAE) capable of producing counterfactual data that are rare in the training dataset. Thanks to the generative model in VAE, we are able to estimate the counterfactual POC directly and use them as part of the optimization goal to learn features with good causal properties.

Acknowledgements

First of all, I would like to express my deepest gratitude to my supervisor, Wanli Ouyang, who has supported me through my PhD with countless discussions and exchanges of ideas. Considering the topics I chose are absolutely cutting-edge, he also introduced me to many other experts who provided me with deep insights and proofread my ideas with their knowledge. Particularly, I would like to express my deepest gratitude to Chaochao Lu, who not only helped develop my ideas, suggest experiments, and analyze the results with me, but also helped to revise my papers and construct rebuttals. This journey would not have been possible without you, and I could not have asked for more incredible people to guide me.

I also want to express my gratitude to the readers of this thesis, for your interest in my topics and for spending time to read them. Whatever your purpose may be, I wish that you find something useful in these pages, and I wish you every success in your endeavors.

Finally, I would like to thank my colleagues at the University of Sydney. We have spent much time together and gone through a lot, especially the difficult time during the COVID. Thank you for the joy that you brought me at those fun organized parties. My special thanks also go to Jessica McBroom, who introduced me to Australia when I first got here and let me tag along to spend Christmas with her and her family, it has been an incredible time.

Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Name: Senhui Guo Signature: _____

Contents

CHAPTER 1

Introduction

1.1 Neural Networks

Neural networks (NNs) have proven to be an extraordinary framework for tackling challenging tasks. They started off as an attempt to imitate biological neural systems, where the key idea is to combine simple elementary computation units (i.e., neurons) into complicated systems capable of handling cognitively difficult tasks. As a result, the NNs can take a lot of forms, some are layers of linear transformation stacked together, some have skipping connections, some have local convolutions, and some have current structures. Thanks to the rapidly improving computation capacity of hardwares, NNs have since gained incredible capabilities to tackle extremely complicated tasks such as protein structure prediction with AlphaFold [19], image generation with the recent diffusion model [8], new transformer backbones [53] for all sorts of frameworks, and Large Language Models like ChatGPT. These tasks are usually out of reach of traditional machine learning methods, primarily because their capability is largely limited by manual human design, which requires a clear understanding of the underlying process. In other words, how close the designed theory behind the model is to the actual process determines how well the model can possibly perform. If the real process is too complicated for us to design an approximating theory, then it's impossible for us to even properly model the process.

The train-test diagram of NNs exempts us from understanding the process in advance. Instead, we can provide the NNs with enough capacity/expressiveness, and then design a target or a loss function to tell NNs what kind of results we are expecting with the ground truth labels. For instance, if we want to train a NN to classify images, every time, we provide the NN with

an image and a label, then pass the image through the network to get the prediction from the NN and then we compare the prediction to the actual label to give a score. Then, the score is used to update the parameters of the NN to make the prediction closer to the label. This process is repeated until the NN can consistently produce correct predictions from input images. When it comes to other forms of tasks, the procedure is basically the same. We give the network a standardized input and tell the network what should be produced on the output end.

However, such excellent usability of NNs tends to obscure their learning procedure, leading to the black-box [2] situation in which one cannot comprehend how the capability for a task is obtained. Unlike traditional methods, where the model follows a theory and each component has a mathematical interpretation, NNs defuse the information into the countless connections between neurons, making it impossible to interpret the model as a whole.

As a result, previous works on adversarial attacks like [34] etc., have shown that NNs are gullible and only require a little manipulation to produce unexpected results. For example, Nguyen et al. [34] have shown that a NN trained to classify images can be easily fooled by adding a small amount of noise to the image. Normally, when a picture of panda was sent to the NN, the prediction would be "Panda". But by adding a carefully designed noise that's imperceptible to human eyes, we can change the prediction to "Dog", which is completely irrelevant. If we can somehow understand how these results are produced inside the network, then maybe we can find a way to prevent certain unexpected behaviors from happening.

Thus, interpreting NNs has remained one of the pressing tasks and a major obstacle to developing general artificial intelligence. Because of the nature of this problem, there are many different approaches to it, including Grad-CAM [49], which attempts to highlight the "important" part of an image to a NN, and LIME [42], which tries to explain the prediction with a simpler function. But all these methods are based on correlations, they might be able to tell us what the NN is looking at, but they can not tell us why the NN is looking at it. Thus we need a new approach to tell us why, and that's where causality comes into play.

1.2 Causality: A New Dimension

The advances in the field of causality have helped us open a new portal to explore things that are intractable under traditional probability-based frameworks. One shining example is the recent works on identifying causal features with multiple environments [1]. We know that spurious features are hard to deal with in traditional probabilistic models. When the spurious features and the actual labels are highly correlated, it's hard to distinguish between the two. Consequently, if you train your model to predict the labels, it might end up learning the spurious features instead of the actual labels. By introducing the concept of causality, and leveraging some causal optimization goals, we're able to reliably identify causal features that are invariant across different environments.

We know that NN structures are hard to identify. Every time we run the training, the structure would be different, and it would be really hard to provide a consistent explanation to the structure. By combining causality into the interpretation of NN structures, we might be able to see through all the variant and correlated parts, and discover what's really invariant inside the model. We believe these invariant structures are the key to understanding NNs.

Pearl [36] systematically studied the "Ladder of Causation" and highlights the distinct roles of seeing, doing and imagining. Seeing refers to association, or correlation, which simply describes the statistical relationship between variables. Doing refers to intervention, which is the act of changing the value of a variable. Other than statistical associations, you need correct causal directions in addition in order to predict correct responses to interventions. Imagining refers to counterfactuals, given a reality that's happened, what would be alternative outcome look like if we had done something differently. Counterfactuals subsume interventional and associational questions, and are the most general form of causal queries.

In the following sections, we will follow the ladder of causation. First, we'll start with traditional probabilistic models. Then, we introduce the concept of intervention and how we can expand the probabilistic model to Bayesian Causal Models to deal with it. Finally, we introduce counterfactuals and structural causal models, which are able to describe counterfactual concepts.

1.3 Probablity and Graphical Models

A general theoretical perspective of NNs has always been probability theory, e.g., the last layer of a classification NN has been regarded as a discrete probability distribution. A general probabilistic system consists of several random variables $\mathbf{X} = \{X_1, \dots X_n\}$, whose distribution can be described as a joint distribution

$$P(\mathbf{X}) = P(X_1, \dots, X_n). \tag{1.1}$$

The joint distribution works fine for a simple system with only several random variables. However, we would face an enormous challenge with such a vanilla representation when we try to use it to describe a more complicated system. As the number of random variables increases, the storage required to store the information of the distribution explodes. To illustrate this, let's assume each random variable is discrete and can take values from a set of K elements. As a result of this, the table required to describe the joint distribution will be of size K^n. For each instance $\mathbf{x} = \{x_1, \dots, x_n\}$, you need a real number to record its probability $p_x = P(X_1 = x_1, \dots, X_n = x_n)$. For continuous distribution, it is going to be even more complicated.

However, if we are given additional information about their conditional independence, we would be able to compress the distribution. For instance, if given $X_i \perp \{X_1, \dots, X_{i-2}\} | X_{i-1}$, which is usually known as the *Markov property*, we can decompose the joint distribution into

$$P(\mathbf{X}) = P(X_1, \dots, X_n) = P(X_1)P(X_2|X_1)P(X_3|X_2) \dots P(X_n|X_{n-1}), \tag{1.2}$$

which is a product of n local conditional distributions. Except for $P(X_1)$ which requires a table of size K to describe, each conditional distribution $P(X_i|X_{i-1})$ only requires a table of size $K \times K$ to describe. As a result, the total size of tables to describe the system is only $K + (n-1)K^2$, which is significantly smaller than K^n if we were to use the full joint distribution.

As we have just shown, the joint distribution of the system can be dramatically compressed with their conditional independence relationships. Generally, a joint distribution $P(\mathbf{X})$ can be

expressed as

$$P(\mathbf{X}) = \prod_i P(X_i|\mathrm{E}_i), \tag{1.3}$$

where evidence $\mathrm{E}_i = \{X_j | j \in I_i\}$ is a set of variables considered as the parents of X_i. We can give this representation a graphical interpretation: if $j \in \mathrm{PA}_i$, then there is an edge pointing from j to i. Further, we can define the compatibility of a distribution to a graph:

DEFINITION 1.3.1 (Compatibility of Distribution to Graph). *We say a distribution P is compatible with a graph G if the distribution can be decomposed into*

$$P(\mathbf{X}) = \prod_i P(X_i|\mathit{PA}_i), \tag{1.4}$$

where PA_i is the parents of X_i in the graph.

Because of the way we decompose the joint probability, the graph that is compatible with a distribution is always acyclic, otherwise, a variable will eventually be the evidence of itself through a cycle of the graph. When we talk about the graphical representation of a probabilistic model, it is almost always a directed acyclic graph (DAG), because without giving additional definition or interpretation, the cycles in the graph will not make any sense in probabilistic terms.

So is there some sort of interpretation of independence in graphs that correspond to probability independence? How do these two seemingly faraway conceptual universes connect?

First, we define a concept called *d-separation* [36] for both paths and triplets X, Y, Z:

DEFINITION 1.3.2 (d-Separation of a Path). *A path p is said to be d-separated or blocked by a set of nodes Z if and only if any of the following conditions are met.*

- *p contains a graphical structure called chain $i \to m \to j$ or a fork $i \leftarrow m \to j$ such that the middle node m is in Z.*
- *p contains an inverted fork or graphical structure called collider $i \to m \leftarrow j$ such that m or its descendants are **not** in Z.*

FIGURE 1.1. **Equivalent Graphical Structure.**

DEFINITION 1.3.3 (d-Separation of X and Y). *X and Y are said to be d-separated by Z if every path from a node in X to a node in Y is d-separated by Z.*

The bridge between graphs and probability lies in the probabilistic interpretation of d-separation.

THEOREM 1.3.4. *Assume we have a DAG G, and (X, Y, Z) are three disjoint subsets of the variables in G. we have:*

- *if X and Y are d-separated by Z, then X and Y are independent given Z for any distribution P that is compatible with G.*
- *if X and Y are independent given Z for **all** distributions P that is compatible with G, then X and Y are d-separated by Z.*

For a distribution P, we have multiple graphs that are compatible with it at the same time. For example, for a two-variable system, X and Y, it is perfectly reasonable to decompose the joint distribution into $P(X|Y)P(Y)$ or $P(Y|X)P(X)$. Even if they are independent, we can still do it as $P(X|Y) = P(X)$ or $P(Y|X) = P(Y)$. The two graphical structures compatible with the two conditional probability decompositions, shown in Figure 1.1, are equally good in terms of expressing independence relationships and make probabilistic predictions. We say that the two structures are observationally equivalent because we can not distinguish the two models with only observational data.

DEFINITION 1.3.5 (Observational Equivalence). *Two DAGs G_1, and G_2, are observationally equivalent if and only if they have the same skeletons and the same sets of v-structures. Skeleton refers to edges regardless of their directions. G_1 and G_2 have the same skeleton if, for any edge from X to Y in G_1, there is either an edge goes from X to Y or Y to X in G_2. V-structure refers to two converging arrows whose tails are not connected by an arrow. This property is also known as Markov equivalence [36].*

Once the graphical interpretation of probability distributions is given, we can interpret NNs as probabilistic systems, which is something that we have been doing for a long time. We have many works that are built upon this idea, including Markov Random Field, Conditional Random Field, Variational Autoencoder, etc.

1.4 Causal Bayesian Models

As we have talked about, a distribution has many compatible graphs G as long as they conform to the decomposition of the joint distribution. Now the question is, are they all really the same, do we ever need to distinguish between them?

A graphical interpretation of the model does not necessarily mean causality, that is, if X has an edge pointing to Y, it does not necessarily mean X is a direct cause of Y given other variables. Let's think about the case for temperature and the reading on a thermometer. Let T be the temperature, and R be the reading. If we do not give any information regarding their dependence, then there are two ways to interpret their relationships, $P(T|R)P(R)$ and $P(R|T)P(T)$. The first one says that the temperature is controlled by the reading on the thermometer (as if there is a magical mechanism so that somehow the thermometer is able to heat up or cool down the air to change the actual temperature), and the reading has its own distribution. The second theory is the one that we know, for a fact, is correct, that the reading on the thermometer is controlled by the actual temperature, and the temperature conforms to its own distribution. For convenience, we assume T and R are perfectly correlated, i.e., T=R, thus

$$P(T|R=r) = \begin{cases} 1 \text{ if } T=r \\ 0 \text{ o.w.} \end{cases} \quad \text{or } P(R|T=t) = \begin{cases} 1 \text{ if } R=t \\ 0 \text{ o.w.} \end{cases} \tag{1.5}$$

These two understandings are equally good when we want to predict the real temperature from the reading on the thermometer. For the first model $P(T|R)P(R)$, the temperature

equals the reading. For the second one $P(R|T)P(T)$, the temperature follows the posterior

$$P(T|R) = \frac{P(T,R)}{P(R)} = \frac{P(R|T)P(T)}{P(R)} = \begin{cases} P(R|T)\frac{P(T)}{P(R)} = 1 \text{ if } R = T \\ 0 \text{ o.w.} \end{cases} . \tag{1.6}$$

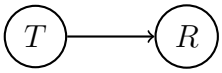
Thus the temperature also equals the reading. The problem emerges when we want to change the actual temperature. For that to happen, the two theories give us completely different suggestions. The first one says that we can simply change the reading on the thermometer by, for example, heating up the thermometer such that the thermometer will cause the temperature to change. The second one says that there is nothing we can do about the temperature if we're only allowed to change the reading on the thermometer unless we are given the tools to directly alter T itself. This is where all of the magic begins. We are trying to directly change a variable, which is not something you can describe in a probabilistic model because probability is all about observations. When we want to discuss the relationship between two random variables T and R, we can only use conditional probability $P(T|R)$ to describe it under the framework of probability models. But because we do not change the model during the discussion of probability, we are not saying how changing the value R will change the value of T. Instead, we are only extracting a subset of the joint distribution where R meets our condition. To distinguish the two concepts, we introduce *do* operation. When we conduct $do(X = x)$, we shield X from all other variables that can directly affect X and set X to be x. We call such operations interventions. Accordingly, a causal Bayesian network is a graphical model that can respond correctly to these interventions:

DEFINITION 1.4.1 (Causal Bayesian Network [36]). *Assume $P(\mathbf{v})$ is a distribution on a set of variables V. $P_x(\mathbf{v})$ is the distribution resulting from the intervention $do(X = x)$ that forcefully set a subset X to be x. We use P_* to denote the set of all interventional distributions $P_x(\mathbf{x})$ for $\forall X \in V$, including the unaltered distribution $P(\mathbf{v})$, where $X = \emptyset$. A DAG G is said to be a causal Bayesian network compatible with P_* if and only if*

- *$P_x(\mathbf{x})$ is compatible with G.*
- *$P_x(\mathbf{v}_i) = 1$ for all $V_i \in X$ whenever $\mathbf{v}_i$ is consistent with $X = x$*

- $P_x(\mathbf{v}_i|PA_i) = P(\mathbf{v}_i|PA_i)$ *for all* $V_i \notin X$ *whenever* PA_i *is consistent with* $X = x$*, i.e., each* $P(\mathbf{v}_i|PA_i)$ *remains invariant to interventions not involving* V_i*.*

Let's see what the response from the model would be if the correct causal graph is given. For temperature and thermometer reading, we know the correct causal graph is T causing R. When we conduct an intervention $do(T = t)$ on T, we remove its connection from its parents. In this case, since T follows its own distribution and has no parents, we do not need to modify the model, as for the reading R, which follows $P_t(R|T = t) = P(R|T = t)$, is still the conditional distribution of R given T. This is easy to understand because T has no parents, so the intervention done on T would be equivalent to conditioning on T. On the other hand, if we conduct an intervention on R, the edge from T to R will be removed as part of the intervention procedure, and thus the temperature will remain at its original value, and this is consistent with the real-life experience that changing the reading on the thermometer does not change the actual temperature. The graphical structure

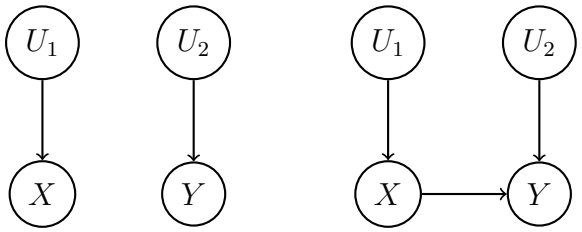


would be the only structure that is compatible with all interventions. Thus it is the causal Bayesian model for the temperature-thermometer problem.

Probability models work perfectly fine if we only constrain our tasks to statistical predictions. But when it comes to directly altering the model, only a graphical model with true causal relationships can give us the correct predictions of the outcome of that change.

1.5 Counterfactual

So, are causal Bayesian models good enough? In real-world situations, we often talk about concepts like "the probability of event B happened *because* of event A" or "the probability that event B would have been different if it were not for event A". These concepts rely on the reasoning about an individual piece of data. We call such statements counterfactual questions, because at least a part of the statement contradicts reality. For a causal Bayesian model,



(A) Model 1 (B) Model 2

FIGURE 1.2. **Two Functional Models With Same Distribution.**

we can discuss an interventional distribution $P(Y|do(X = x))$, which is the probabilistic property of Y when X is forcefully set to x. But this interventional distribution is used to describe the population behavior of Y. If given a data sample x' and y', we can not use interventional distribution to describe what Y would have been if X was set to other values other than x'. This is because once a sample has been taken from the joint distribution, we do not have the apparatus to change or manipulate it anymore in the traditional probabilistic graphical model.

However, we can improve this by using functional relationships as well as random exogenous variables. Let's take a look at an example where causal bayesian models fail. Assume we have the simplest causal Bayesian network consisting of a pair of independent binary variables X and Y. Let X stand for the event that a patient receives treatment ($X = 1$), and Y be the event that a patient recovers ($Y = 1$). Assume now we have a patient, who has taken the treatment ($X = 1$) and recovers ($Y = 1$). We want to investigate this individual case and find out if his recovery occurred **because of** the treatment, **despite** the treatment, or **regardless of** the treatment. Assume we have two models

Model 1 (Figure 1.2a)

$$\begin{aligned} X &= U_1 \\ Y &= U_2, \end{aligned} \tag{1.7}$$

where U_1 and U_2 are two independent variables with binary values such that

$$P(U_1 = u) = \begin{cases} 0.5 & \text{if} \quad u = 0 \\ 0.5 & \text{if} \quad u = 1 \end{cases} \qquad P(U_2 = u) = \begin{cases} 0.5 & \text{if} \quad u = 0 \\ 0.5 & \text{if} \quad u = 1 \end{cases} \tag{1.8}$$

For model 2, we have

Model 2 (Figure 1.2b)

$$\begin{aligned} X &= U_1 \\ Y &= XU_2 + (1-X)(1-U_2), \end{aligned} \tag{1.9}$$

where U_1 and U_2 follow the same probability as model 1. If we lay out the joint distributions for the two models as Table 1.1, we found that their joint distributions are identical. It means that the models would behave exactly the same in terms of probability distributions.

Now let's go back to the question that we are interested in, had the patient not taken the treatment, what would happen to him? The given conditions, which are facts that have already happened, are $X = 1$, and $Y = 1$. For model 1, taking treatment and the final outcome are two independent events, we can infer that $U_1 = X = 1$ and $U_2 = Y = 1$. But since the occurrence of Y is only dependent on U_2, even if we are given a second chance to stop him from receiving the treatment, i.e. set $X = 0$ forcefully, the fact that $U_2 = 1$ is still going to have the patient recovered anyway. Thus the answer from model 1 to our question would be, had the patient not taken the treatment, he would have recovered regardless. For model 2, it is a bit confusing if we directly look at the functional relationships. A way to interpret the model is that U_2 indicates the type of body constitution. For patients of type 0 ($U_2 = 0$), they would definitely die if they are given the treatment as $Y = (1 - X)$, and would definitely recover if they are **NOT** given the treatment. On the other hand, patients of type 1 ($U_2 = 1$) would definitely die if they are **NOT** given the treatment and would definitely recover if they receive the treatment. Assume model 2 is the true underlying model, for our patient, since we know he received the treatment ($X = 1$) and recovered ($Y = 1$), we can infer that his body constitution type is $U_2 = 1$. Thus, had he not received the treatment ($do(X = 0)$), the new outcome

$$Y' = X'U_2 + (1 - X')(1 - U_2) = X'U_2 = 0 \cdot 1 = 0 \tag{1.10}$$

indicates that the patient would have died.

We have constructed two models with exactly the same marginal distribution and the same causal Bayesian network since the graph $X \to Y$ is compatible with the joint distribution,

Model 1	$U_2 = 0$		$U_2 = 1$		Joint distribution	
	$X = 1$	$X = 0$	$X = 1$	$X = 0$	$X = 1$	$X = 0$
$Y = 1$ (Recovery)	0	0	0.25	0.25	0.25	0.25
$Y = 0$ (Death)	0.25	0.25	0	0	0.25	0.25

(A)

Model 2	$U_2 = 0$		$U_2 = 1$		Joint distribution	
	$X = 1$	$X = 0$	$X = 1$	$X = 0$	$X = 1$	$X = 0$
$Y = 1$ (Recovery)	0	0.25	0.25	0	0.25	0.25
$Y = 0$ (Death)	0.25	0	0	0.25	0.25	0.25

(B)

TABLE 1.1. **Joint Distribution for the Two Functional Models.**

but they produce completely different answers to our questions. It shows that models based on purely stochastic processes are insufficient for calculating the counterfactual outcome, therefore, we need a more expressive model that is constructed with functional relationships.

1.6 Structural Causal Models

In the previous section, we have shown that functional models are more expressive in the sense that they can express counterfactual statements more precisely. Structural causal models (SCM) are the most common causal framework, which can be described in the functional form as $M = \langle \mathbf{U}, \mathbf{V}, \mathbf{F} \rangle$, $\mathbf{U}$ being the exogenous variables, which are independent from each other; $\mathbf{V}$ being the endogenous variables, $f_i \in \mathbf{F}$ being the mapping from $U_i \cup \text{PA}_i$ to V_i where $\text{PA}_i \subset \mathbf{V}$ is the parent nodes of V_i. The graphical relationships are defined in $\mathbf{F}$ in the sense that, for any V_i and V_j, if V_j is a variable in f_i, then V_j is a parent of V_i. An SCM requires the corresponding graph to be a DAG. The SCM can be expanded [37] as

$$V_i = f_i(\text{PA}_i, U_i; \Theta_i), \tag{1.11}$$

where $V_i \in \mathbf{V}$, Θ_i is the parameter for f_i.

To conduct counterfactual reasoning, we follow the procedures: Given an SCM, e is the current evidence, we want to know what would Y have been if X were x.

- **Step 1 (abduction):** Update the probability $P(U)$ with $P(U|e)$
- **Step 2 (action):** Intervene and set $X = x$ by shielding X from the influence of its parents.
- **Step 3 (prediction):** Use the modified SCM to compute the probability of Y as $P(Y_x|e)$

To formulate a NN under the SCM framework, each node is considered a causal variable; the (weights of) neural links and the activation functions deciding the value of each individual node are causal mechanisms, like F_i in an SCM. A single node alone does not help much to interpret a complex model since they need to work with each other in a bunch to achieve specific utilities, like handling context information. Thus, we need to discover modular structures consisting of these nodes to provide more sophisticated interpretations.

1.7 Research Questions

The ultimate goal is to set up the network so that when the data distribution shifts, the network can adapt to the new distribution "smartly". We approach this goal from two different angles: active and passive. On the active side, in Chapter 2, we decompose the network into several causal modules, and prepare multiple modules trained on different data distributions for substitution. Whenever the data distribution changes, we can intervene and replace the modules with the ones that are trained on the new distribution. On the passive side, in Chapter 3, we want to learn a causal representation that is invariant to the data distribution, which only consists of factors that are causally related to the label, so that when the distribution changes, the representation remains the same, and generalize well to out-of-distribution samples.

CHAPTER 2

Modular Structure Discovery in NNs

In this chapter, we propose a structural mechanism identification method, namely, *competitive disentanglement*. Specifically, for models from two different tasks but sharing common aspects, we divide each model into two components: *a common mechanism*, which is a mechanism (i.e., a set of neurons) shared by both models, and an *individual mechanism*, which is a mechanism exclusive to each respective model. When we force the two models to share (the parameters of) part of their neurons, neurons from the common mechanism will outcompete the ones from the individual mechanisms in terms of maximizing performance because they do not need to compromise their functions between the two models. As a result, by observing how sharing neurons impact the performances of the two models, we can discover neurons from the common mechanism and thus disentangle NNs into modules dedicated to different functions. We provide extensive experiments to show that competitive disentanglement is capable of identifying interpretable mechanisms in NNs implemented as SCMs, and these mechanisms can be further reused in downstream tasks.

In the rest of the chapter, we first provide an intuitive introduction, explaining the motivation and the basic idea that we used in our method along with an example of training networks on MNIST and KMNIST. We also review related works dedicated to network disentanglement and modularity. After that, we formally introduce the formulation of our model, quasi-SCM, and provide methods for training the model. Then we introduce the framework to identify mechanisms in our model. Finally, we conduct experiments on various datasets to demonstrate that our model is capable of discovering modular structures in NNs.

2.1 Introduction

In this chapter, we explore the possibility of identifying modular structures in NNs using causal frameworks. According to the Principle of Independent Mechanisms [47], there are autonomous causal mechanisms/modules inside NNs. If we can discover them, then we might be able to intervene in the network and make it adapt to new tasks by conducting appropriate manipulation.

One natural idea to interpret NNs is to break them into smaller pieces, as complicated systems usually consist of smaller modules. Tasks like domain adaptation suggest that there are autonomous modules inside NNs that can be reused [61, 52]. However, it is extremely difficult to explore such modularity in NNs under the current framework, where NNs are constructed from basic building blocks (e.g., convolutional or linear layers). It is apparent that these building blocks do not exhibit such modularity because each can be involved in multiple modules. This idea inspires us to resort to structural causal models (SCMs) [36, 39]. One advantage of SCMs is that they offer better interpretability and manipulability by describing a system as a combination of mechanisms (i.e., autonomous modules), which is an ideal framework for exploring modularity.

Take for example the MNIST [23] classification. It is conceivable that a NN would require one mechanism to identify symbols and another to handle context information (e.g., background and style). For convenience, we call the former *the symbol-identifying mechanism* and the latter *the context-handling mechanism*. If we now turn to KMNIST [6], which is a dataset of handwritten Japanese characters, a corresponding NN would need to posses the same context-handling mechanism since the images from KMNIST share the same type of context information with those from MNIST, only the symbols are different. Theoretically speaking, if the mechanisms in a NN could be separated, then transferring the NN learned from MNIST to KMNIST would be more efficient. We can directly transfer the context-handling mechanism, and only need to re-learn the symbol-identifying mechanism on KMNIST. However, before we can take advantage of these mechanisms, we need to know where these mechanisms

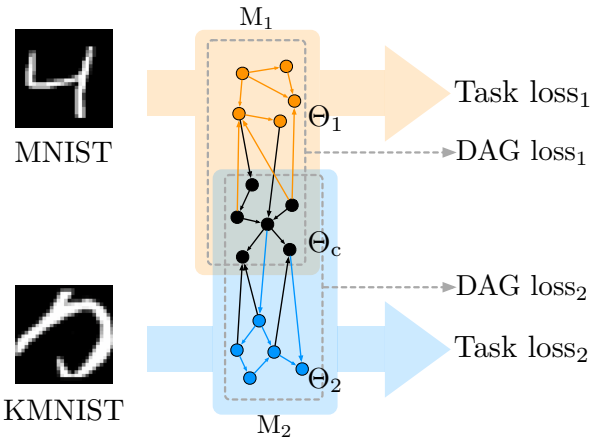


FIGURE 2.1. **Mechanism identification via competitive disentanglement.** We first set up our NN models with the all-to-all connection among nodes, which enables them to automatically learn to decide which nodes/parameters to share. Then two models M_1 and M_2 sharing the parameters Θ_c are trained on two datasets MNIST and KMNIST simultaneously with DAG losses to guarantee acyclicity. In this case, the parameters of their common mechanism will be shared before those of their individual ones, because sharing common parameters does not impact performance (cf. Lemma 2). However, after sharing all the parameters of their common mechanism, they will start sharing the parameters of their individual mechanisms, which will affect their performances (cf. Lemma 1). Hence, by observing varying performance as the number of their shared parameters changes, we can deduce the size as well as the specific parameters of their common mechanism.

are, i.e., which nodes/neurons[1] these mechanisms consist of, respectively. This is not to say that SCMs can help us automatically find where these nodes are, which is the problem our framework tries to tackle, but they do offer better flexibility, unlike NNs where nodes are packed together into layers.

2.1.1 Competitive Disentanglement

To illustrate our idea of competitive disentanglement for mechanism identification, we further investigate the two NNs used for the dual classification tasks on MNIST and KMNIST, as shown in Figure 2.1. Our goal is to identify both *the common mechanism* for context-handling, which is shared across the two tasks, and *the individual mechanisms* for symbol-identifying,

[1]We use node in our paper to refer to the functional relationship from the parents of the node to the node itself, instead of the value of the node.

which are different across the two tasks. If each of the two NNs is forced to choose which node to share and the chosen node belongs to the common mechanism, then the performance of each NN on their respective task will not be affected. The reason is that the chosen node belonging to the common mechanism play the same role in both tasks. However, if the chosen node does not belong to the common mechanism, then the performance of either NN would be undermined. This is because this node has to take a trade-off between the tasks on MNIST and KMNIST. Assume that we have an oracle optimizer, which is capable of finding which node to share in each NN in order to achieve the best performance. Now, we can start to seek for all the nodes belonging to the common mechanism by gradually increasing the number of nodes being forced to share. At the early stage, the performance won't be affected because there are plenty of nodes from the common mechanism to choose from. However, once the number required to be shared exceeds *the size of the common mechanism*, then the optimizer has to include part of the individual mechanisms, and the nodes from that part are going to make compromises, when the performance starts to drop. As such, the size of the common mechanism is the largest number of nodes being shared without reducing performance. The NN trained under the largest number are separated into three parts: the two individual mechanisms respective for MNIST and KMNIST (cf. Θ_1 and Θ_2 in Figure 2.1), and the common mechanism (Θ_c), which is the set of common nodes that the optimizer chooses to share. It is also worth mentioning that the identified common mechanism is solely determined by the two datasets without any subjective criteria. We call this approach to identifying the common mechanism *competitive disentanglement*.

2.1.2 A Model Without Structural Implications

As aforementioned, the optimizer has to be capable of finding the most suitable nodes to share to achieve best performance. For a regular NN, the optimizer has to iterate over all possible sets of nodes and then choose the one that produces the best performance. This is practically impossible because of the exponential curse. Motivated by this difficulty, we propose the quasi-SCM, a graphical model that allows all-to-all[2] connections, where the

[2]In this paper, for a graph to have all-to-all connection means to be able to have a connection between any two nodes, it does not necessarily mean that any two nodes has to have a connection.

connection relationships are encoded into an adjacency matrix. Note that, a regular NN can perfectly fit into such a model since it is essentially a graph. Thanks to the all-to-all connections, it implies that all nodes are positionally equivalent in quasi-SCMs. Hence, if the optimizer needs to select l nodes to share, it can, without loss of generality, choose the first l nodes (implemented as the first l columns in the adjacency matrix), and learn the graphical connections as the training goes on. In other words, the initial selection of nodes to share does not matter. This can save us from the exponential curse of iterating we encounter in the regular NN. This requires that the optimizer has the full freedom to adjust any connections in the graph. For this, we allow for cycles in the graph, as loops are inevitable during the adjustment. However, this creates two issues: 1) it does not satisfy the requirement of the directed acyclic graph (DAG) we aim for; 2) the values of nodes can no longer be calculated in topological order. To address them, we propose to leverage fixed-point dynamics to get the solution for the quasi-SCMs with cycles included in the graph. To guarantee the DAG-ness of the graph at the end of the training, we propose *additive DAG loss*, which is linearly scalable to the size of the graph, to ensure that the graph will eventually converge to a DAG.

2.1.3 The Role of Causality

It is important to point out that competitive disentanglement is not designed to conduct causal discovery, nor does it guarantee genuine causal directions among individual nodes. Our focus is to discover the existing modular structure in a graphical system. Although on the module level, the competitive disentanglement can be regarded as looking for mechanisms that can minimize an interventional goal: $D_{KL}(P(Y_1|do(C_1 = c_2), X)|P(Y_1)) + D_{KL}(P(Y_2|do(C_2 = c_1), X)|P(Y_2))$, where Y_1, Y_2 are the labels from the two datasets, X_1, X_2 are the high-dim data and C_1, C_2 are the functional relationships and c_1, c_2 are the parameters of those relationships. This is also related to works of causal abstraction. With the help of this interpretation, we are able to give fine-tuning of NNs a causal meaning, which is the retraining of the individual mechanisms. But competitive disentanglement does not guarantee causal connections within the module. In fact, although KMNIST can be considered as an out-of-distribution test of MNIST, but if you put them under the perspective of common mechanism

vs individual mechanism, there are three models learning from three i.i.d distributions, i.e., the common mechanism learning from the common aspects from KMNIST plus MNIST and two individual mechanisms from the individual aspects of the two datasets respectively. Therefore, there is no distribution shifts happening during the training, thus how faithful the functional structure of these modules are to the actual data processing precedure is irrelevant to our task, which is why we didn't describe them as causal mechanism in our paper. As long as those modules are working properly even with potential spurious connections, they won't affect our task of transfer learning, where we intent to transplant/manipulate modules in a whole. The reason we adopted the SCM for formulation is that it is a very general framework where causal manipulations are well defined so that we can change modules under a rigorous framework, but as for how how causally accurate the structrue is depends on how the SCM is trained, which is not within the scope of our work.

Contributions: 1) We propose to characterize mechanisms by exploiting the common aspects across multiple datasets. 2) We propose to extend SCMs to quasi-SCMs that can tolerate cycles within the graph, and use competitive disentanglement with quasi-SCMs to identify the characterized mechanism. 3) We propose additive DAG loss to reduce cyclicity of the graphs of quasi-SCMs. 4) We use extensive experiments to verify the disentanglement and show that the disentangled mechanism can be used in downstream tasks.

2.2 Background

2.2.1 Neural Network Interpretability

Interpreting NNs has been an important topic and develops quickly [64, 11, 28]. Although NNs have achieved high performance on many tasks, even outperforming humans in some cases, they still have great disadvantages when it comes to explainability. This is almost inevitable if you consider the humongous number of independently working neurons/nodes inside a NN, just like we can not understand how human brains work at the moment. Of course, the fact that NNs are built as a collection of computer operators gives us the chance

to conveniently study and make tweaks to the system, while biological neural systems are not nearly as available as NNs are. Such complexity directly leads to many unexplained and unexpected behaviors. Szegedy et al. [51] has shown that by inserting an imperceptible noise on the input image, you can force a trained NN to produce an arbitrary prediction result. In reverse, Nguyen et al. [34] developed a method to construct a noise-looking image that is completely unrecognizable to humans to pass through the trained NN and produce any desired prediction with very high confidence. Although people have tried many methods to increase the generalization ability of NNs, these examples show that if we want to rely on NNs for critical tasks, we have to "understand" them more.

Naturally, there are two pathways in front of us, either we try to understand how a NN is trained and gained its ability to handle the task, or we add more implications to the network to manually guide the formation of modularity/interpretability. The former is referred to as passive methods [9, 55] and the latter is active methods [62, 40]. Passive methods usually approach interpretability by studying how a bunch of nodes affect another bunch of nodes, but how these elementary mechanisms form into something bigger or higher level is still a major obstacle. It is easy to find out the direct cause(s) of a particular prediction result, but trying to understand how these results are produced from low-level pixels to high-level abstractions with the help of structural mechanisms in a NN is another thing (we use structural mechanisms to generally refer to functional relationships that are not necessarily causal). Active methods, on the other hand, incorporate human understanding into a NN to form modularity. If you take this more broadly, convolutional neural networks can also be considered as manually designed modular NNs. When designing the convolutional layer, people impose the structural implication that pixels should be processed locally as groups of pixels, and such processing modules should share the same logic/parameters, which is why nowadays we have convolutional kernels. Another example is the modular NNs [21] where networks are assembled through the combination of selected modules. These methods although improved the modularity of NNs, and consequently exhibit better manipulability, fail to offer explanations for how NNs acquired their abilities by themselves. You may give as much instruction on modularity as you want, but there will always be part of NNs that is still a black box. There is no doubt that there are modules inside NNs, and we can induce better

modularity by explicitly constructing NNs in a modular way, but we will not be able to put our trust in NNs if we can not understand how modularities are autonomously acquired by NNs.

2.2.2 Causality for Interpretation

Causal models following Pearl's graphical formulation [47] are a popular framework to deal with non-i.i.d. data that NNs are struggling to handle. A causal model can be decomposed into multiple mechanisms [39] instead of a monolithic mapping like NNs, which offers us a window to discover how data are processed inside NNs. Furthermore, modular decomposition empowers us to apply interventions [3] to NNs to handle unseen data. Previous work has closely studied the properties of causal mechanisms [50, 4, 35, 16] and how they can be utilized [46, 3]. Others have also explored the possibility of combining causal models with NNs. For example, Ke et al. [20] replaces the mapping in SCMs with NNs. Wiering [58] add causal relations into NNs to obtain better generalization. Xia et al. [59] introduce a class of trainable SCMs. However, few of them have explored how mechanisms in NNs can be directly characterized and identified.

Parascandolo et al. [35] proposed a method that is particularly related to our work. They first constructed a diversified MNIST data where the original MNIST images are additionally processed with various transformers to apply noise, distortion, etc. Then they line up a list of MLPs whose output is also an image of the same size and designed a selection mechanism so that whenever an image is given to the model, the model will first decide which MLP it goes to, and then the output, which is a new processed image will be fed into a regular classification network. After the model training is completed, they found that these MLPs have independently learned separate detransformers to recover the transformed MNIST images back to the original form. It shows that NNs are able to form natural modularity under competition. However, this work relies on subnetworks to achieve modularity, quite like modular NNs, we still do not know what is really going on inside a network. This inspires us to create a competitive environment for single neurons instead of subnetworks to compete

FIGURE 2.2. **Independent Causal Mechanism.** When the causal direction in the model is correct, the intervention applied to one mechanism will not affect others.

in a task. If such a task is carefully designed, then it is possible to acquire modularity in the form of neurons.

2.2.3 Independent Causal Mechanism Principle

Apart from Pearl's theory, the idea of discovering modules or mechanisms is inspired by the Independent Causal Mechanisms Principle [48]. It states *The causal generative process of a system's variables is composed of autonomous modules that do not inform or influence each other. In the probabilistic case, this means that the conditional distribution of each variable given its causes (i.e., its mechanism) does not inform or influence the other mechanisms.* A common example to demonstrate this idea is the distribution of altitude and temperature. Given a set of binary data (A, T) for a list of cities, A is the altitude of that city, and T is the temperature. From the perspective of graphical models, we have two perfect explanations or theories for the data as shown in 2.2. First, we have $P(T|A)P(A)$, where we believe A of cities follows a distribution while T is governed by a conditional distribution $P(T|A)$. Of course, we know this is the correct explanation as $P(T|A)$ is the physical mechanism that determines the temperatures when located at different altitudes. However, we can also think of the model as $P(A|T)P(T)$, where we think temperatures T follow a distribution, and altitudes are governed by a conditional distribution $P(A|T)$ given the temperature. These two explanations are perfectly fine when we only deal with data that are i.i.d. However, if the climate has changed, and these cities are experiencing rising temperatures, then we do not need to change $P(A)$ as the altitudes of cities have not changed, the only part we need to adjust is $P(T|A)$ since the resulting temperatures have risen because of the climate. As a result, the new theory $P'(T|A)P(A)$ would explain the new data. However, if we look at the second theory, because the distribution of temperatures has changed, so we need a new $P'(T)$

to explain the new temperatures. At the same time, the altitudes have not changed, and the conditional $P(A|T)$ will also need to change to fit the new theory. Thus, we would have to alter all components of the model to $P'(A|T)P'(T)$ to befit the new data. As we can see, if the causal relationships are correct, the interventions that happened to one mechanism will not spill over to other mechanisms, and we only need to update the intervened mechanism if we want to make the model fit the new data once more. This principle guides how we should define disentanglement and how we should approach modularities in complicated systems. Inspired by this law, the disentanglement and identification of mechanisms also refer to the independence of functional relationships instead of the statistical independence of random variables.

2.2.4 Cyclic Causal Models

SCMs that allow for circles are known as cyclic SCMs, which are usually adopted when feedback loops are involved. There are a number of works extensively studying the cyclic SCMs [5, 13, 29, 12, 30, 32]. In our case, cyclic SCM is only an intermediate stage, and no additional definition of cycles is given. Such a case can be regarded as a fixed point problem [31], and the solution can be obtained through a recursive method. Another topic that carries some resemblance is the implicitly defined layer in NNs [63].

In comparison, the advantage of our method is that we use causal theory as our bedrock, which offers a well-established framework for interpretation. In addition, our method can expose the most original form of mechanisms consisting of nodes.

2.3 Quasi-SCMs

Before we address quasi-SCMs, we need to add some specifications to the SCMs and how an SCM can fit into a discriminative task. An SCM can be described as

$$V_i = f_i(\mathrm{PA}_i, U_i; \Theta_i). \tag{2.1}$$

In a general inference setting, a part of $\mathbf{V}$ denoted as $\mathbf{V}_I$ will be given as $\mathbf{v}_I$, and the goal is to find the MAP (Maximum A Posteriori) estimation of $\mathbf{U}_{-I} = \mathbf{U} \backslash \mathbf{U}_I$ and (consequently) $\mathbf{V}_{-I}$ such that the likelihood is maximized, i.e., $\hat{\mathbf{U}}_{-I} = \arg\max_{\mathbf{U}_{-I}} P(\mathbf{U}_{-I} | \mathbf{V}_I = \mathbf{v}_I)$. $\mathbf{V}_{-I}$ can be calculated from $\hat{\mathbf{U}}_{-I}$ and $\mathbf{v}_{-I}$. For example, if this SCM describes a discriminative model of classification, then $\mathbf{V}_I$ would be the input data, and a part of $\mathbf{V}_{-I}$ would be the prediction results, denoted as $\mathbf{V}_O(\mathbf{V}_I; \Theta)$.

Parameter sharing has not been new for NNs, e.g., Sachan and Neubig [45] proposes to share a subnetwork in the model to improve performances. The distinction between our quasi-SCM and other layer/block-sharing method [44] is that we identify mechanisms on the neuron level, i.e., we put every node under scrutiny whether it belongs to the targeted mechanism. For this, we extend the acyclic graphs of SCMs to cyclic ones with potential all-to-all connections among the nodes. These graphical relationships are parameterized by an adjacency matrix to provide sufficient granularity to the optimizer to adjust connections via gradients. Under the quasi-SCM framework with a cyclic graph, if we were to force two models to share l nodes, without loss of generality, we could appoint the first l nodes (i.e., the first l columns in the adjacency matrix) in the two models to share their parameters. This is because all nodes are equivalent and interchangeable under the framework. Formally, we extend SCM to quasi-SCM as

$$V_i = f_i(\mathbf{PA}_i, U_i; \Theta_i) = f(W_{\cdot i}^T \mathbf{V}, U_i; \theta_i), \tag{2.2}$$

where the graphical relationships are represented by the adjacency matrix $\mathbf{W} = (W_{\cdot 1}, \ldots, W_{\cdot n}) \in \mathbb{R}^{n \times n}$, $W_{\cdot i} = (W_{1i}, \ldots, W_{ni})^T$, f is the universal mapping function parameterized by θ_i, $\Theta_i = \theta_i \cup W_{\cdot i}$. The corresponding graph is no longer required to be a DAG, and hence we call it quasi-SCM. We call a function or a set of functions deciding the value of a neuron or that of a set of neurons a mechanism.

DEFINITION 2.3.1 (Mechanism). *In a quasi-SCM, the functional mapping from a neuron's parents to the neuron, i.e. $f_i(\cdot) = f(\cdot; \Theta_i)$ is called an elementary mechanism. A mechanism is a set of elementary mechanisms, containing at least one elementary mechanism.*

Note that an elementary mechanism is also a mechanism. From this perspective, a fully connected layer in a NN is a mechanism, and so is a convolutional layer. Because each node is connected with all the other nodes, the graph is no longer required to be a DAG. That is, quasi-SCM might include cycles in the graph, which disqualifies the model to be an SCM, hence the name quasi-SCM. As pointed out, all causal relationships are parameterized, and all nodes have the same parameter space, thus they are equivalent in the sense that they have the same potential to learn any mechanisms. We also use a DAG loss $h(\mathbf{W})$ as a penalty term to enforce acyclicity during optimization so that our model can eventually be represented as an SCM and be able to use causal tools for SCMs. Note that, because we will use the quasi-SCM to model the classification task, we decide to discard the random exogenous variables $\mathbf{U}$ and make the model deterministic as involving random factors during the inference will not help with predictions; Thus, the exogenous variables $\mathbf{U}$, namely the source of randomness, are ignored. The problem can be formulated as

$$\min_{\Theta} \text{ Loss}(\mathbf{v}_I, \mathbf{v}_O) = d(\mathbf{V}_O(\mathbf{v}_I; \Theta), \mathbf{v}_O) \quad \text{subject to} \quad h(\mathbf{W}) = 0, \tag{2.3}$$

where $\mathbf{v}_I$ is the input data, $\mathbf{v}_O$ is the ground truth labels, $\mathbf{V}_O$ is the predictions inferred from the input and the parameters $\Theta = \cup_i \Theta_i \cup \mathbf{W}$, $d(\cdot, \cdot)$ is the loss function (e.g., cross-entropy loss), and $h(\mathbf{W})$ is the DAG loss measuring acyclicity of the graph with the adjacency matrix W such that whenever $h(\mathbf{W}) = 0$ the graph is acyclic.

2.3.1 Solving quasi-SCMs

To elaborate on $\text{Loss}(\mathbf{v}_I, \mathbf{v}_O)$, we require additional specifications about quasi-SCMs. Each node $\mathbf{v}_i \in \mathbb{R}^d$ is a vector $(v_{i1}, \ldots, v_{id})$. Unlike SCMs where nodes can be calculated in a topological order from roots to leaves, evaluating quasi-SCMs embedded in cyclic graphs can be viewed as a fixed-point problem [31] and thus can be solved via recursive iteration. It is crucial that we make sure the gradients do not decay in the recursive process. Thus, we select ReLU [33] as the activation function following a linear transformation. We implement the

mapping from PA_i to $\mathbf{v}_i$ as $\mathbf{v}_i = f(W_{\cdot i}^T \mathbf{V}; \theta_i) = F(L_i(\sum_j W_{ij}\mathbf{v}_j + \mathbf{b}_i))$, where F is ReLU, $L_i \in \mathbb{R}^{d\times d}$ a linear transformation, $\mathbf{W}$ the adjacency matrix, and $\mathbf{b}_i \in \mathbb{R}^d$ the bias term. Our goal is to find a solution $\mathbf{V}^*$ for $\mathbf{V}$ such that the above equation holds true for $\forall \mathbf{v}_i \in \mathbf{V}$. Once the solution is acquired, we can proceed to calculate the classification loss. For fixed-point problems, we have the following theorem to acquire solutions.

THEOREM 2.3.2 (Banach fixed-point theorem [14]). *Let (X, d) be a non-empty complete metric space with a contraction mapping $T : X \to X$ such that $d(T(x) - T(y)) < d(x, y)$ for $\forall x, y \in X$, then T admits a unique fixed point x^* such that $T(x^*) = x^*$. Furthermore, x^* can be found as follows: start with an arbitrary element $x_0 \in X$, and define a sequence $(x_n)_{n\in\mathbb{N}}$ where $x_n = T(x_{n-1})$, then $x^* = \lim_{n\to\infty} x_n$.*

Apply Theorem 2.3.2 to our quasi-SCM, we have the following results:

PROPOSITION 2.3.3 (Existence of fixed-point for quasi-SCMs). *For a quasi-SCM, we claim that whenever $||W||_1 \leq 1$, $||L_i||_1 \leq 1$ for $\forall i$, $||\cdot||_1$ being the L^1 norm, the mapping is contractive and thus the fixed point exists and can be found via the Banach fixed-point theorem.*

PROOF. With T denoting the mapping and $||\cdot||_1$ being the metric, We have

$$\begin{aligned} |T(V\mathbf{v}_1) - T(\mathbf{v}_2)| &= |\mathbf{F}(\mathbf{W}^T\mathbf{V}_1 + \mathbf{B}) - F(\mathbf{W}^T\mathbf{V}_2 + \mathbf{B})| \\ &= \sum_i |f_i(W_{\cdot i}^T\mathbf{v}_1 + \mathbf{b}_i) - f_i(W_{\cdot i}^T\mathbf{v}_2 + \mathbf{b}_i)| \\ &= \sum_i |F(L_i(W_{\cdot i}^T\mathbf{v}_1 + \mathbf{b}_i)) - F(L_i(W_{\cdot i}^T\mathbf{v}_2 + \mathbf{b}_i))| \end{aligned} \tag{2.4}$$

where $W_{\cdot i} = (W_{1i}, W_{2i}, \ldots)^T$, F is the ReLU function. For each term, we denote $L_i(W_{\cdot i}^T\mathbf{v}_1 + \mathbf{b}_i)$ as c_1 and $L_i(W_{\cdot i}^T\mathbf{v}_2 + \mathbf{b}_i)$ as c_2, because of the property of ReLU and the additivity of the

linear transformation L_i, we can easily deduce

$$|f_i(c_1) - f_i(c_2)| = \begin{cases} |c_1 - c_2| & c_1 \geq 0,\ c_2 \geq 0 \\ |c_1| < |c_1 - c_2| & c_1 \geq 0,\ c_2 < 0 \\ |c_2| < |c_2 - c_1| & c_1 < 0,\ c_2 \geq 0 \\ 0 & c_1 < 0,\ c_2 < 0 \end{cases} \tag{2.5}$$
$$\leq |c_1 - c_2| = |L_i(W_{\cdot i}^T(\mathbf{v}_1 - \mathbf{v}_2))|$$

Finally, we have

$$\begin{aligned} |T(V\mathbf{v}_1) - T(\mathbf{v}_2)| &= \sum_i |f_i(W_{\cdot i}^T\mathbf{v}_1 + \mathbf{b}_i) - f_i(W_{\cdot i}^T\mathbf{v}_2 + \mathbf{b}_i)| \\ &\leq \sum_i |L_i(W_{\cdot i}^T(\mathbf{v}_1 - \mathbf{v}_2))| \\ &\leq \sum_i ||L_i||_1 \cdot |W_{\cdot i}^T| \cdot |\mathbf{v}_1 - \mathbf{v}_2| \\ &\leq (\sum_i ||L_i||_1 \cdot |W_{\cdot i}^T|)|\mathbf{v}_1 - \mathbf{v}_2| \\ &\leq (\sum_i |W_{\cdot i}^T|)|V_1 - V_2| = ||W||_1 \cdot |\mathbf{v}_1 - \mathbf{v}_2| \leq |\mathbf{v}_1 - \mathbf{v}_2| \end{aligned} \tag{2.6}$$

□

Thus, as long as the adjacency matrix and the linear transformations are maintained such that $||W||_1 \leq 1$, $||L_i||_1 \leq 1$ for $\forall i$, the solution exists and can be found by the recursive mapping as

$$\mathbf{v}_i^* = \lim_{k\to\infty} \mathbf{v}_i^k \tag{2.7}$$

where $\mathbf{v}_i^{k+1} = \lim_{k\to\infty} F(L_i(\sum_j W_{ji}\mathbf{v}_j^k + \mathbf{b}_i))$.

2.3.2 Scalable DAG loss

There are a few papers [65, 66] working on using differentiable DAG loss to characterize the acyclicity of the graph. Zheng et al. [65] proposed such a loss as $h_e(\mathbf{W}) = \text{tr}(e^{\mathbf{W}\circ\mathbf{W}}) - d$, where $\mathbf{W} \in \mathbb{R}^{d\times d}$ is the adjacency matrix of a cyclic graph, $\circ$ the Hadamard product, and e^A

the matrix exponential of $\mathbf{A}$, i.e., $e^{\mathbf{A}} = \sum_{n=0}^{\infty} \frac{1}{n!}\mathbf{A}^n$. We use *exponential DAG loss* to refer to $h_e(\mathbf{W})$. Since the graph is acyclic whenever the exponential DAG loss is zero, we can use it as the optimization goal and let the optimizer use gradients to control the acyclicity of the graph.

However, this exponential DAG loss fails when applying to large graphs. Assume we have a **simple cycle** in the graph, which is a finite sequence of n nodes $v_1 \to v_2 \to \cdots \to v_{n+1}$ connected by n edges $e_1, e_2, \ldots, e_n$, where $v_{n+1} = v_1$ and e_i from v_i to v_{i+1}. The contribution of this cycle to the exponential DAG loss $h_e(\mathbf{W})$ is $\frac{1}{n!}e_1^2 e_2^2 \ldots e_n^2$. Thus, in practice, the contribution of a cycle to the loss rapidly decreases to zero as the length increases. To make quasi-SCMs work on practical image datasets, we are motivated to propose a loss which can linearly scale to the size of the graph.

We define an operator $\bar{\cdot} : \mathbb{R} \to \{0, 1\}$ that can be applied to a matrix $\mathbf{A}$, such that

$$\bar{A}_{ij} = \begin{cases} 1 \text{ if } A_{ij} \neq 0, \\ 0 \text{ o.w.} \end{cases} . \tag{2.8}$$

Now, we propose a new loss as $h_a(\mathbf{W}) = \text{tr}(|\mathbf{W}| \sum_{n=0}^{\infty} \bar{\mathbf{W}}^n)$, where $|\mathbf{W}|$ is the absolute of $\mathbf{W}$, i.e., $|W|_{ij} = |W_{ij}|$.

PROPOSITION 2.3.4 (Linearly scalable DAG loss). *If there is a simple cycle c: $v_1 \to v_2 \to \cdots \to v_{n+1}$ of length n in graph G, where $v_1 = v_{n+1}$, and $\mathbf{W}$ is the adjacency matrix, the contribution of this cycle to the DAG loss $h_a(\mathbf{W})$ is $h_c = \sum_{i=1}^{n} |e_i|$, where e_i is the edge from v_i to v_{i+1}.*

PROOF. Since cycle c is simple, the contribution of c is independent from other cycles, thus we assume these's no other cycles in the graph WLOG. Because $\bar{W}_{ij}$ indicates whether there is a direct path connects from node i to node j, it's obvious that $\bar{W}^k_{ij} \in \{0, 1\}$, which is the (i, j) entry of matrix $\bar{W}^k$, indicates whether there is a path that travels from node i to node j after k steps. For $\forall v_i \in c$, $(\bar{W}^{n-1})_{v_{i+1}v_i} = 1$ because there's a path $v_{i+1} \to \ldots \to v_n \to v_1 \to \ldots \to v_i$ that travels $n - 1$ steps from v_{i+1} to v_i, also because it's a simple cycle,

there's no other path from v_{i+1} to v_i. Thus

$$\begin{aligned}(|W|\bar{W}^{n-1})_{v_iv_i} &= \sum_j |W|_{v_ij}(\bar{W}^{n-1})_{jv_i} \\ &= |W|_{v_iv_{i+1}}(\bar{W}^{n-1})_{v_{i+1}v_i} \\ &= |e_{v_i}|.\end{aligned} \tag{2.9}$$

Hence

$$\begin{aligned}h = h_a(W) &= \text{tr}(\bar{W}^{n-1}|W|) \\ &= \sum_{j=1}(\bar{W}^{n-1}|W|)_{jj} \\ &= \sum_{v_i\in c}(\bar{W}^{n-1}|W|)_{v_iv_i} \\ &= \sum_{i=1}^{n}|e_i|\end{aligned} \tag{2.10}$$

□

As long as the weights of the edges remain smaller than an upper bound as in the contraction condition in Proposition 2.3.3 where $e_i \leq ||\mathbf{W}||_1 \leq 1$ for $\forall i$, we have $h < \sum_i^n 1 = n$. Thus, the value of our loss grows linearly with the length of the cycle. We refer to our loss as *additive DAG loss*. It is worth mentioning that if the cycle is not simple, edges within the cycle may contribute multiple times to the loss. However, once the graph is sparsified during the training, the simple cycle assumption holds true to the extent that the crossings of non-simple cycles will not affect the linear scalability of the loss, which is verified in our experiments.

2.3.3 Solving DAG constraint

Once the additive DAG constraint is given, we can use augmented Lagrangian method [17, 65] to handle the constraint for the optimization problem, which turns the objective (2.3) to

$$\begin{aligned}\min_{\Theta} \quad & \text{Loss}(\mathbf{v}_I, \mathbf{v}_O) + \frac{\rho}{2}|h_a(\mathbf{W})|^2 \\ \text{subject to} \quad & h_a(\mathbf{W}) = 0,\end{aligned} \tag{2.11}$$

where Θ is the parameters of the model, including W, B and L_i, $\mathbf{v}_I$ and $\mathbf{v}_O$ are the input and target data respectively. The advantage of augmented Lagrangian is that it approximates the solution of a constrained problem by the solution of an unconstrained problem. Essentially, the algorithm can be written as a dual ascent method. The dual function with Lagrange multiplier is given by

$$D(\alpha) = \min_{W} L^{\rho}(W, \alpha) \tag{2.12}$$

$$\text{where} \quad L^{\rho}(\mathbf{W}, \alpha) = \text{Loss}(\mathbf{v}_I, \mathbf{v}_O) + \frac{\rho}{2}|h_a(\mathbf{W})|^2 + \alpha h_a(\mathbf{W}). \tag{2.13}$$

We need to find a local solution to the dual optimization problem $\max_{\alpha \in \mathbb{R}} D(\alpha)$.

Let $\mathbf{W}^*_\alpha$ be the local minimizer of $D(\alpha)$, α can be updated by $\alpha \leftarrow \alpha + \rho h_a(\mathbf{W}^*_\alpha)$.

$$\alpha \leftarrow \alpha + \rho h_a(W^*_\alpha). \tag{2.14}$$

The choice of learning rate ρ can be guided by the following theorem:

THEOREM 2.3.5. *For ρ large enough and the starting point α_0 near the solution α^*, the update 2.14 converges to α^* linearly.*

2.4 Mechanism Identification

2.4.1 The optimality of the solution of quasi-SCMs

In order to have a theoretical discussion about the impact of parameter sharing on performance, we have to assume that gradient descent is able to find the solution to achieve global optimal performance. In this section, we study to what degree this assumption holds true.

There have been some theories analyzing how well gradient descent, which is almost the exclusive optimization method that people use what it comes to NN training, finds the global minima for the loss target. Du et al. [10] provides a set of conditions under which gradient descent is able to obtain zero training loss.

THEOREM 2.4.1 (Convergence Rate of Gradient Descent for MLPs). *Assume for all* $i \in [n]$, $||\mathbf{x}_i||_2 = 1$, $|y_i| = O(1)$ *and the number of hiden nodes per layer*

$$m = \Omega\left(2^{O(H)} \max\{\frac{n^2 \log(\frac{Hn}{\delta})}{\lambda_{\min}^2(\mathbf{K}^{(H)})}, \frac{n^4}{\lambda_{\min}^4(\mathbf{K}^{(H)})}, \frac{n}{\delta}\}\right), \tag{2.15}$$

where $K^{(H)}$ *is the Gram matrix. If we set the step size*

$$\eta = O(\frac{\lambda_{\min}(\mathbf{K}^{(H)})}{n^2 2^{O(H)}}), \tag{2.16}$$

then with probability at least $1 - \delta$ *over the random initialization, we have, for iteration* $k = 1, 2, \ldots$

$$L(\theta(k)) \leq \left(1 - \frac{\eta \lambda_{\min}(\mathbf{K}^{(H)})}{2}\right)^k L(\theta(0)). \tag{2.17}$$

It basically states that when the activation function is smooth, the width of each layer is large enough, and optimization steps are appropriately planned, then the risk minimization loss $L = ||f(\mathbf{x}) - \mathbf{T}||_2$, where f is the NN, $\mathbf{x}$ is the input data, $\mathbf{T}$ is the target label, converges to zero at a linear speed.

For the activation function, we can use smooth ReLU without undermining the purpose of not decaying the gradients during recursive mapping. The only major gap here is that our NN is embedded in a potentially cyclic graph which does not fall into the definition of NN provided by Du et al. [10]. However, we can show that as long as our graph is close to the domain of the acyclic graph, the global optimal solution would also be close by.

First, we give the definition of the distance between a cyclic graph and the acyclic domain.

DEFINITION 2.4.2 (Distance to Acyclic). *Let* g *be a cyclic graph, and* $G_a = \{g_a | h_a(\mathbf{W}_{g_a}) = 0\}$, *where* $\mathbf{W}_g$ *is the adjacency matrix of graph* g, *is the set of all acyclic graphs. The distance between* g *and* G_a *is defined as*

$$d(g, G_a) = \min_{g_a} ||\mathbf{W}_g - \mathbf{W}_{g_a}||_1. \tag{2.18}$$

It is easy to see that when $d(g, G_a) < \epsilon$, the Gram matrix of the NN embedded in g and g_a would also be close enough, i.e., $||\mathbf{K}_g^{(H)} - \mathbf{K}_{g_a}^{(H)}||_1 < C\epsilon$. Thus the loss function at each

iteration satisfies

$$L(\theta(k)) \leq \left[\left(1-\frac{\eta\lambda_{\min}(\mathbf{K}^{(H)})}{2}\right)^{k} + k\left(1-\frac{\eta\lambda_{\min}(\mathbf{K}^{(H)})}{2}\right)^{k-1} C'\epsilon + o(\epsilon^2)\right] L(\theta(0)). \tag{2.19}$$

Thus as long as the distance between our cyclic graph and the acyclic domain is close enough, the convergence rate for our model is also linear when relevant requirements are satisfied. Thanks to our linear DAG loss, our model can reach acyclic in linear time, and the distance between the graph of a quasi-SCM and the acyclic domain will be maintained at a very low level.

2.4.2 Identification via competitive disentanglement

Given datasets $\mathcal{D}_1$ and $\mathcal{D}_2$, the common mechanism shared by the two models for these two datasets will be our target mechanism to identify.

DEFINITION 2.4.3 (Model Equivalence). *For two sets of parameters of quasi-SCMs, we say that $M = \langle V, F\rangle$ and $M' = \langle V', F'\rangle$ are equivalent up to a permutation, if there exists a permutation p, such that $f_i = f'_{p(i)}$, and consequently $\mathbf{v}_i = \mathbf{v'}_{p(i)}$.*

Because of the all-to-all connection, any optimal solution that maximizes the loss function has a class of equivalent solutions that can also maximize the performance. We can divide all parameters in terms of nodes as $\Theta = \cup_{i\in V}\Theta_i$, Θ_i being the parameters of the mapping f_i. Assuming the tasks have optimal solutions, we denote the unique global optimal solutions to (2.3) on the two datasets up to equivalence by $\Theta^{*\dagger}$ and $\Theta^{*\ddagger}$, respectively. For the common mechanism, we have the following assumption.

ASSUMPTION 2.4.4 (Existence of common mechanism). *Assuming there is a common mechanism sharing parameters for two quasi-SCMs. Due to the equivalence, there exists an $l^* \in \mathbb{N}$ and two permutations $p^\dagger$ and $p^\ddagger$, such that, $\Theta^{*\dagger}_{p^\dagger(i)} = \Theta^{*\ddagger}_{p^\ddagger(i)}$ for $i \in \{1,\ldots,l^*\}$, and $\Theta^{*\dagger}_{p^\dagger(i)} \neq \Theta^{*\ddagger}_{p^\ddagger(i)}$ for $\forall i > l^*$.*

Next, we will use competitive disentanglement to get l^*. For a given integer value l, we can optimize two models in parallel while sharing l nodes during the training. For ease of reference, we call these two models *parallel models*. Since all nodes are positionally equivalent under the quasi-SCM, we can designate the **first** l nodes to be shared without loss of generality. The overall optimization problem can be formulated as

$$\begin{aligned} \min_{\Theta^\dagger,\Theta^\ddagger} \quad & \text{Loss}(\mathbf{v}_I^\dagger, \mathbf{v}_O^\dagger; \Theta^\dagger) + \text{Loss}(\mathbf{v}_I^\ddagger, \mathbf{v}_O^\ddagger; \Theta^\ddagger) + \frac{\rho}{2}(|h_a(W^\dagger)| + |h_a(W^\ddagger)|)^2 \\ \text{subject to} \quad & h_a(W^\dagger) = \ h_a(W^\ddagger) = 0, \ \Theta_i^\dagger = \Theta_i^\ddagger \text{ for } i \in \{1, 2, \ldots, l\}. \end{aligned} \tag{2.20}$$

For simplicity, we denote $\text{Loss}(\mathbf{v}_I^\dagger, \mathbf{v}_O^\dagger; \Theta^\dagger) + \text{Loss}(\mathbf{v}_I^\ddagger, \mathbf{v}_O^\ddagger; \Theta^\ddagger)$ as $\overline{\text{Loss}}_l$, and the minimal loss using optimal parameters as $\overline{\text{Loss}}_l^*$, we also denote $\min_l \overline{\text{Loss}}_l^*$ as $\underline{\text{Loss}}$. In terms of the performance of the two quasi-SCMs on the two datasets with regards to the number of shared nodes l, we have the following two conclusions:

LEMMA 1 (monotonicity). *For $l^* < l_1 < l_2$, then $\overline{Loss}_{l^*}^* < \overline{Loss}_{l_1}^* \leq \overline{Loss}_{l_2}^*$.*

PROOF. For $l_1 > l^*$, $\nexists I_1$, $|I_1| = l_1$ and permutation p_1, p_2 such that $\Theta_{p_1(i)}^{*\dagger} = \Theta_{p_2(i)}^{*\ddagger}$ for $\forall i \in I_1$ as there are at most l^* nodes that share the same parameters between the two optima. Thus the two parallel models can not reach optimal solutions at the same time when l_1 nodes are required to be shared, thus $\overline{\text{Loss}}_{l^*}^* = \text{Loss}_{l^*}^{*\dagger} + \text{Loss}_{l^*}^{*\ddagger} < \text{Loss}_{l_1}^{*\dagger} + \text{Loss}_{l_1}^{*\ddagger} = \overline{\text{Loss}}_{l_1}^*$.

The second part is obvious because requiring $l_2 > l_1$ nodes to be shared places more restrictions than l_1, thus will not produce better results than l_1. □

This lemma indicates that when the number of nodes required to be shared l_i is larger than the size of the common mechanism l^*, the performance of the two quasi-SCMs will be lower than their optimal performance, and increasing l_i will lead to further decreased performance.

LEMMA 2 (Upper bound). *For $l \leq l^*$, $\overline{Loss}_l^* = \overline{Loss}_{l^*}^*$.*

PROOF. Because there exists a common mechanism of size l^*, for the global optima $\Theta^{*\dagger}$ and $\Theta^{*\ddagger}$, there exist l^* nodes sharing exactly the same parameters, as $\exists\ I^*$, $|I^*| = l^*$ and

permutation p such that $\Theta^{*\dagger}_{F_i} = \Theta^{*\ddagger}_{F_{p(i)}}$ for $i \in I^*$. Thus when the number of shared nodes $l \leq l^*$, we can always assign the shared nodes with the nodes from the common mechanism and fill the parameters of the rest of the nodes with the parameters from the optimal solutions. Thus the two quasi-SCMs can reach their optimal solutions together, and the sum of losses for l will be exactly the same as l^*, which is $\overline{\text{Loss}}^*_{l^*}$.

□

This lemma indicates that when the number of shared nodes l is equal to or smaller than the size of the common mechanism l^*, the performances will be the same as l varies.

From the conclusions above, we can deduce the size of the common mechanism:

THEOREM 2.4.5. *The size of the common mechanism is*

$$l^* = \max\{\arg\min_l \overline{Loss}\}. \tag{2.21}$$

PROOF. From Lemma 2 and Lemma 1, we know that $\overline{\text{Loss}}^*_{l^*} = \underline{\text{Loss}}$, which is the optimal performance, thus $l^* \in \arg\min_l \overline{\text{Loss}}$. Also from Lemma 1, l^* is the largest value l can reach without increasing the loss, thus it is obvious that $l^* = \max\{\arg\min_l \overline{\text{Loss}}\}$. □

The size of the common mechanism l_i is the largest number of nodes being shared without reducing performance. Proofs for these results are in the appendix. As a result, we can sweep the number of shared nodes l from zero up and train the parallel models for each l, until we can notice a performance drop when l surpasses the size of the common mechanism. Once we obtain the size of the common mechanism l^* through competitive disentanglement, the common mechanism of the two quasi-SCMs is $\{F(\cdot; \Theta^{*\dagger}_i) \mid i \in \{1, 2, \ldots, l^*\}\}$.

Previous work has been baffled by the definition of mechanisms because their subjective description leads to ambiguity. The mechanism identified from competitive disentanglement is solely determined by the two datasets without any subjective criteria. Therefore, we argue that one of the major benefits of competitive disentanglement is that it can characterize a mechanism objectively.

2.4.3 Revisit the Cyclic Graph Form

The need to introduce potential cyclic graph first emerges when we want to share N nodes between the two models, for which we have a more intuitive explanation in the second last paragraph in the introduction. The goal is to find a set of nodes to share that has the minimal impact on the performances. For regular NNs, nodes in different positions have different structural implications, for instance, nodes directly following the image input are more suitable to do pixel-level processing, while nodes in the output layer are more suitable for high-level tasks like classification. Since they are intrinsically different, you have to test over all possible selections of nodes before you can find the most suitable set of nodes to share, which is practically impossible because, for example, if the NN has 100 nodes, and you want to share 20 nodes, then there are C_{100}^{20} possible selections.

However, if we use a potential cyclic graph to embed the model, then all nodes in the graph are positionally equivalent, meaning that they have the exact same learning capability. Since the model can adjust the graphical structure during the training, a node that directly follows the image at the beginning may end up as an output node, so it does not matter if you select these 20 nodes or those 20 nodes to share; you will get the same performance results. As a result, if we want to force the model to share 20 nodes, we no longer need to worry about how to select 20 nodes. Instead, we can appoint the first 20 nodes to share without loss of generality, which is why in equation 2.20, we can force the first l nodes to share w.l.o.g., which is something that you can't do in a forward NN. But eventually, cyclicity caused by these adjustments is not what we desire. Thus we use a DAG loss to force the graph back to the acyclic state.

2.5 Experiments

We conduct a series of experiments to validate our theory. In 2.5.1, we show that quasi-SCM has a similar learning capacity as NNs. In 2.5.2, we show that our additive DAG loss is a better DAG constraint than the exponential DAG loss. In 2.5.3, we visualize the disentangled mechanisms. In Section 2.5.4, we provide evidence of disentanglement within the models.

TABLE 2.1. Performances on CIFAR10

	Accuracy	Model Size
MLP_1	0.5028 ± 0.0031	1.3MB
MLP_2	0.5598 ± 0.0044	6.3MB
MLP_s	0.5244 ± 0.0046	2.9MB
AlexNet	0.7451 ± 0.0125	218MB
GNN	0.5690 ± 0.0069	3.0MB
BNN	0.5240 ± 0.0053	13MB
quasi-SCM	$\mathbf{0.6873} \pm 0.0030$	**2.50 MB**

Finally, in 2.5.5, we demonstrate that the mechanisms can be utilized in downstream tasks. All experiments are conducted on a machine with 4 RTX 3080 GPUs. In order to make the turning point of the accuracy plot in Figure 2.4 obvious enough, the number of total nodes is selected such that the capacity of the model is not yet oversaturated. Our quasi-SCM is not sensitive to hyperparameters like learning rate or batch size; the SGD is adopted as the optimizer.

2.5.1 Model Capacity Comparison with NNs

Our motivation is to interpret NNs, but quasi-SCMs are not, by definition, traditional NNs. However, we argue that quasi-SCMs are equally capable as regular NNs, and hence the interpretation of quasi-SCMs mirrors a similar one for NNs. To show the capacity of quasi-SCMs, we compare the performance of our quasi-SCM with that of NNs and other related methods on CIFAR10. From Table 2.1, we can see that our model performs a lot better than MLPs under roughly the same amount of parameters. While the performance of our model is not as high as AlexNet, but the number of parameters in AlexNet is tremandously higher than ours (218MB vs. 2.5MB). Therefore, we conclude that quasi-SCMs have a capacity that is no worse than NNs, given the same amount of parameters allowed to use.

MLP_1 is a regular multi-layer perceptron (MLP), with the layer structure of $3 \times 32 \times 32 \to 500 \to 200 \to 10$; MLP_2 is also a regular MLP, but with a wider and deeper structure; MLP_s is an MLP that has a skip structure to simulate the residual block. GNN is a graph neural network that has a DAG of 225 nodes; the DAG for GNN is randomly generated each time

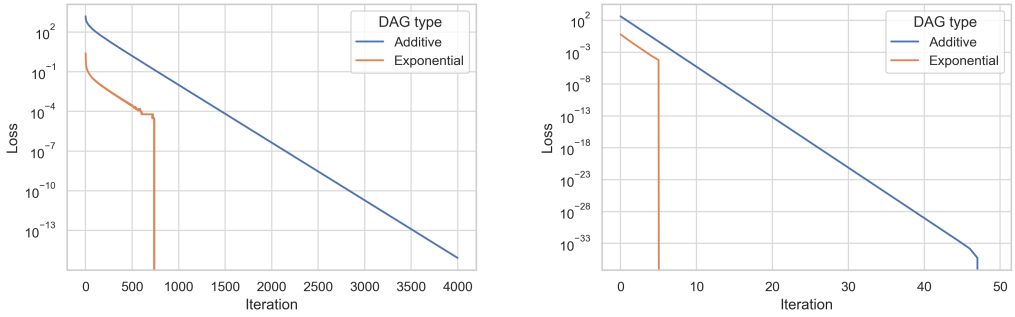


FIGURE 2.3. Comparison of different DAG losses. (a) is trained with our additive loss, $lr = 0.001$, (b) is trained with exponential DAG loss, $lr = 0.1$.

the network is trained. BNN is a bayesian neural network that inherits the structure of MLP_1 but replaces all linear layers with bayesian linear layers.

2.5.2 DAG Loss Comparison

We also compare the performance of our additive DAG loss $h_a(\mathbf{W})$ with that of the exponential DAG loss $h_e(\mathbf{W})$. We use the two losses to train a randomly generated directed cyclic graph (DCG), and monitor the training process with both losses. As shown in Figure 2.3, our loss has a better detection sensitivity (as our additive loss does not easily degenerate to zero, indicated by the vertical line in the figures) and can reduce the DCG to a DAG much faster. Also, because the exponential loss is too insensitive to detect small values of DAG loss, the loss value vanishes a lot earlier than our additive DAG loss.

2.5.3 Mechanism Identification

In this section, we show the direct results of competitive disentanglement, i.e., which nodes form the common mechanism and which form the individual mechanism.

2.5.3.1 Identification Results

Different nodes in the same model play different roles, where some are more critical than others. When uncritical nodes from the individual mechanisms are forced to be shared, the

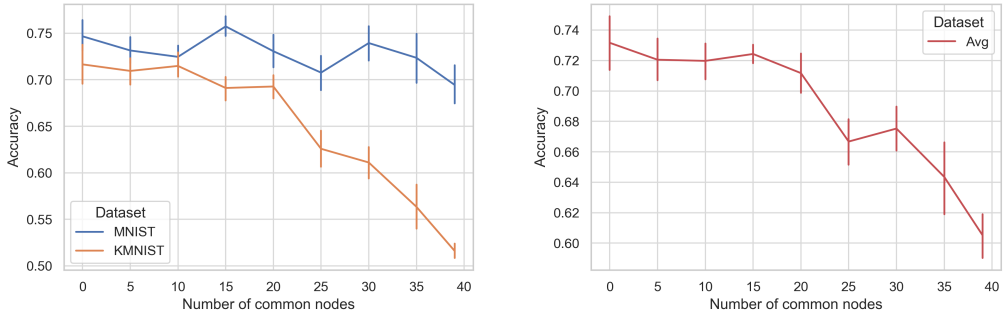


FIGURE 2.4. Accuracy plot to identify the size of the common mechanism (CM). When the number of nodes shared by two quasi-SCMs is smaller than the size of the CM, the accuracies will not be jeopardized as they can share the parameters from the CM. Once the number exceeds the size of the CM, models will have to share nodes from the individual mechanisms, which will affect accuracies. From the plot, we can conclude the size of the CM is 20.

impact they bring is not going to be as significant to the performance as the critical ones. As a result, if we have many such uncritical nodes in the model, the performance drop during competitive disentanglement will not be as obvious as when all nodes carry critical functions. In this case, it may cause difficulty in determining the size of the common mechanism. To avoid this, we need to ensure that the model is not overparameterized.

We select MNIST and KMNIST to conduct the experiment of identifying mechanisms. After competitive disentanglement, the performance against the number of shared nodes is shown in Figure 2.4, from which we conclude that the size of the common mechanism is the turning point 20, where the performance begins to drop. As a result, the common nodes trained under the condition that 20 nodes are shared is the common mechanism that we aim to identify. The rest of the nodes in each model is the individual mechanism.

The reason we believe that our model can perform so much better than MLPs is that our model allows self-organized network structures. The model can form its skips or local convolution if needed, and thus the upper bound of the performance of our model would be a lot higher than vanilla MLPs.

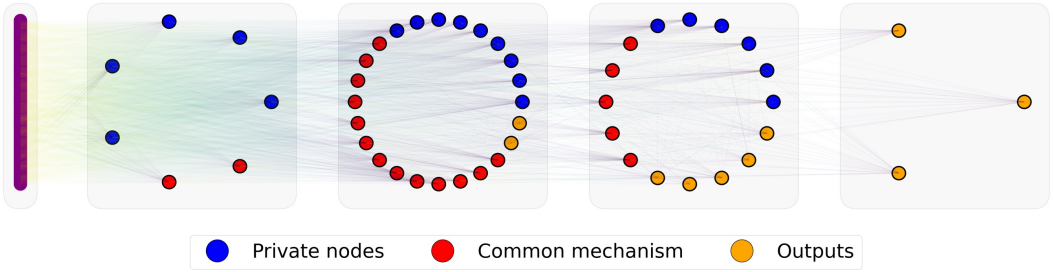


FIGURE 2.5. Mechanism visualization. Inputs are colored purple, blue nodes are the individual mechanism, red nodes are the common mechanism, and orange nodes are the outputs.

TABLE 2.2. Partial disentanglement

Letters	1st-5th	6th-10th	11th-15th
quasi-SCM$_1$	0.8075	0.8685	0.1768
quasi-SCM$_2$	0.7818	0.0962	0.8615

2.5.3.2 Visualizations

We visualize the graph of the quasi-SCM, as shown in Figure 2.5. Different colors represent different mechanisms they belong to. These nodes are divided into layers, on which they converge their values (i.e., $\mathbf{v}_i^k$, which is the intermediate value of nodes during the recursive mapping, as in equation 2.7). The edges are colored such that the color near the tail is yellow and the color near the head is blue.

2.5.4 Evidence of Mechanism Disentanglement

2.5.4.1 Partial disentanglement

The disentanglement of the common mechanism on MNIST and KMNIST may not be evident because we do not have the appropriate tools to directly analyze and verify if they are indeed the mechanisms we are looking for, i.e., whether the common mechanism is indeed in charge of context information, and whether the individual mechanisms are for identifying symbols.

One way to directly observe mechanism disentanglement is through the disentanglement of the symbol-identifying mechanism. By letting two datasets share part of the labels (as well

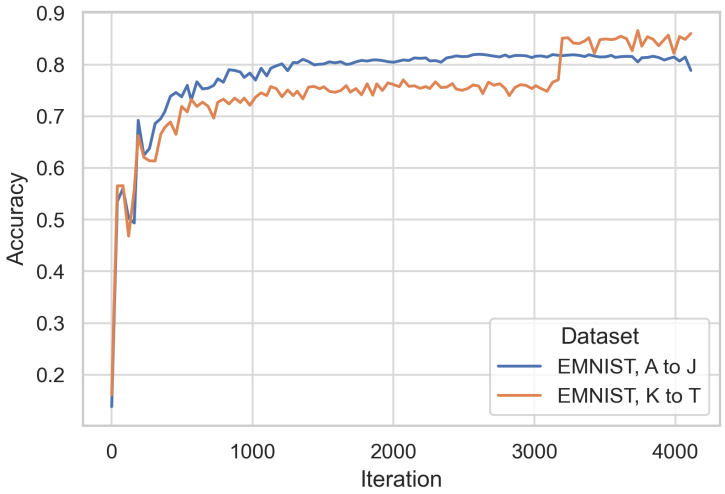


FIGURE 2.6. Disentanglement through unbalanced training. A surge of accuracy on one dataset does not affect the performance of the other.

as the data corresponding to the labels), we can further disentangle the symbol-identifying mechanisms, and the disentanglement will be reflected in the accuracies of those shared and exclusive data. We construct two datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ from EMNIST [7], which is a dataset of handwritten alphabetical letters, and then conduct competitive disentanglement on them. Specifically, $\mathcal{D}_1$ includes the 1st-10th letters, and $\mathcal{D}_2$ includes the 1st-5th and the 11th-15th letters. Thus the overlapping symbol identifying mechanism for the 1st-5th letters is also considered as part of the common mechanism in addition to the context-handling mechanism and will be disentangled into the common mechanism during competitive disentanglement. If our framework can indeed disentangle mechanisms, then we expect the symbol-identifying mechanism for 1st-5th letters to be disentangled into the common mechanism, the symbol-identifying mechanism for 6th-10th disentangled into the individual mechanism for the model on $\mathcal{D}_1$, and the symbol-identifying mechanism for 11th-15th disentangled into the individual mechanism for $\mathcal{D}_2$. As a result, the model on $\mathcal{D}_1$ will not be able to identify the 11th-15th letters, and the model on $\mathcal{D}_2$ will perform terribly when it comes to the 6th-10th letters. The results are exactly what we anticipate, as shown in Table 2.2.

2.5.4.2 Unbalanced training

Unbalanced training is another phenomenon indicating disentanglement. We set up a competitive disentanglement for two quasi-SCMs on two datasets. By feeding different amounts of data during competitive disentanglement, one quasi-SCM model will be prioritized to be the dominant model and get higher performance, while the other will be temporarily left behind to be the secondary model. However, since the capacities of the two models are the same, the secondary model will eventually catch on. Hence, we can observe a surge of accuracy on the secondary quasi-SCM, while the accuracy of the dominant quasi-SCM is not affected, as shown in Figure 2.6. This shows that the common part is not altered during the surge and thus bears little relevance to the individual aspects in terms of mechanism functioning, which demonstrates the disentanglement.

2.5.5 Knowledge Transfer

Since we cannot directly evaluate the disentangled mechanisms alone, we show how the disentangled structure can affect knowledge transfer, specifically, how the fixation of the common mechanism can affect model performance.

2.5.5.1 Mechanism reuse

This experiment shows how the mechanism can be re-used to facilitate training. We set up two quasi-SCMs for MNIST classification, both of which are assigned with the same values of the parameters of the common mechanism but with the rest of the parameters randomly initialized, respectively. We train the first quasi-SCM without constraints. By contrast, we fix the common mechanism of the second quasi-SCM and only train the rest. The comparison shown in Figure 2.7 suggests that fixing the common mechanism can accelerate the training at the early stage.

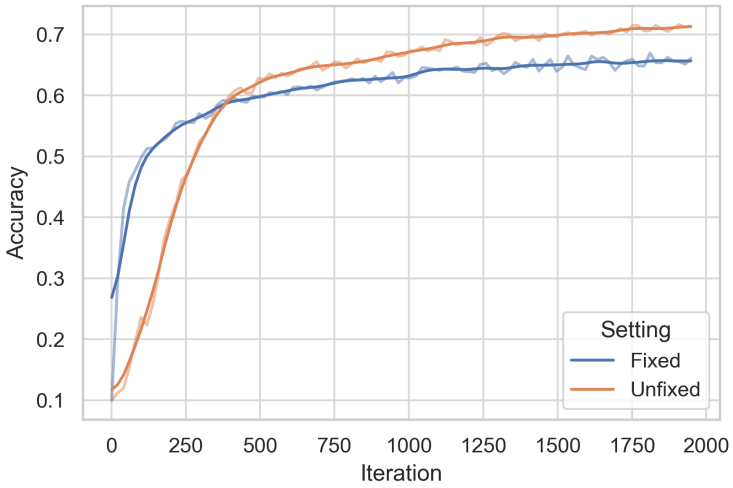
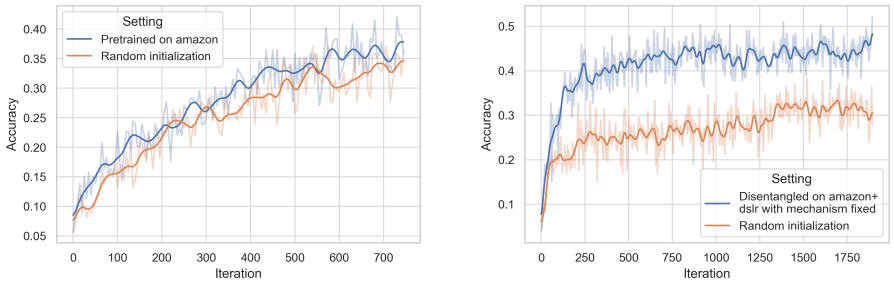


FIGURE 2.7. Mechanism-assisted training. Both quasi-SCMs are assigned with the common mechanism with the rest of the parameters randomly initialized. The mechanism for one quasi-SCM is fixed, and the other is not. The result shows that the mechanism structure helps the training at the early stage.



(A) MLP (B) quasi-SCM

FIGURE 2.8. Comparison of transfer learning ability between MLP and quasi-SCM. For MLP, one is randomly initialized, the other is pretained on *amazon*; For quasi-SCM, one is also randomly initialized, the other is assigned with the common mechanism and fixed. Accuracies are tested on the target domain *webcam*.

2.5.5.2 Transfer Learning

In this section, we show how the identified mechanism can help us transfer models to new datasets. First, we apply our approach to transfer learning to see how the mechanism can help the model improve performance on a new dataset, where models have access to the full

TABLE 2.3. Domain Adaptation on Office-Caltech (SURF)

(A) Different domains

	$D \to A$	$D \to C$	$W \to A$	$W \to C$	Avg.
MLP	0.4323	0.3644	0.3906	0.3511	0.3846
SCM	**0.4384**	**0.5625**	**0.4531**	0.3378	**0.4480**

(B) Similar domains

$D \to W$	$W \to D$	Avg.
0.7627	0.8125	0.7876
0.7119	0.8125	0.7622

dataset of target domain. While in the experiments of domain adaptation, limited data are provided for model fine-tuning.

For transfer learning, the dataset we used is *Office-31* [22]. The source domain is *amazon*, and the target domain is *webcam*. The results are shown in Figure 2.8. We first compared the performance of two MLPs on the target domain *webcam* with one randomly initialized and the other pretrained on the source domain, as shown in Figure 2.8a. Second, we compared the performance of two quasi-SCMs on the target domain with one also randomly initialized and the other assigned with the common mechanism identified on *amazon* and *dslr* fixed, as shown in Figure 2.8b. We can see that using the pretrained parameters as well as maintaining the mechanism structure can help us transfer knowledge more efficiently than simply using the pretrained parameters alone.

For domain adaptation, we use *Office-Caltech* dataset [15]. The dataset contains 4 sub-datasets: *amazon*, *caltech*, *dslr* and *webcam*, denoted as A, C, D and W in Table 2.3 respectively. To test the performance, we first train the model on the source domain, and when adapting to the new domain, we extract 3 images from each label for the model to fine-tune. For quasi-SCMs, we first identify the common mechanism on all four datasets via competitive disentanglement. Then, when the quasi-SCM is fine-tuned on the target domain, the common mechanism is fixed. The results are shown in Table 2.3. Compared to MLP, our model can get better performances on different domains (Table 2.3a) while maintaining performances on the other similar domain (Table 2.3b). The reason is that quasi-SCMs can utilize the common mechanism structure and only fine-tune the individual part, which would be easier compared to fine-tuning the whole model in MLPs. Note that, because the domains are quite different, the identified common mechanism is relatively small, which limits the amount of knowledge we can bring to the new domain with the common mechanism.

2.6 Conclusion and Discussion

In this chapter, we proposed a framework to discover modular structures in NNs. For a single dataset or a data distribution, the definition of a module is always subject to personal views. For example, for the task of MNIST classification, you can argue that the background is handled by a module, and the distortion is handled by a different module. But it is also perfectly reasonable to say that the background and distortion are handled by a single module. It is hard to draw the lines of which function belongs to which modules. We proposed to use multiple tasks and exploit their difference to characterize these modules and further disentangle the whole network into two modules, the individual mechanism and the common mechanism. To identify these modules, we created a competitive environment between two NNs on two different tasks where neurons compete to be the ones that are most compatible with both tasks. We call this competitive disentanglement. The winning neurons are grouped to be the common mechanism and the rest are categorized into the individual mechanism since they only work on their individual task and are not compatible with the other one. To facilitate such competition, we proposed a class of NN called quasi-SCM based on structural causal models, in which the optimizer has the apparatus to pick out the most compatible neurons for both tasks. One side effect of such formulation is that training will inevitably introduce cycles into the graphical structure, hence we further proposed additive DAG loss whose numerical properties enable it to be linearly scalable when encountering large graphs. Further experiments on how the performance changes verified our theory about competitive disentanglement. Experiments on tasks of transfer learning also demonstrate that mechanisms identified using our method are modular in the sense that they can be used in further downstream tasks.

People may ask, even if you can decompose a NN into a common mechanism and an individual mechanism, these two are still black-box NNs just smaller than the overall NN. We believe the significance of this work is that we provide a methodology to break a complicated neural structure down into submodules only relying on objective criteria. When there is only one dataset, there is no need to talk about modularity, NNs are already pretty good at fitting to a singular distribution. It is when we encounter new datasets (whether it is a completely

unrelated dataset with common aspects or an out-of-distribution dataset that is shifted from the original one) that we need to consider modifying part of the NN to adapt to the new distribution. Theoretically speaking, any two different distributions define a common and an individual mechanism, by carefully introducing or constructing new datasets with shifted distribution, we may be able to break the NN structure further down into a set of substructures that reflect the differences between these distributions.

CHAPTER 3

Causal Representation Learning on Image Classification Task

In this chapter, we proposed a framework to learn causal representation with a novel VAE capable of conducting counterfactual generation. It is quite common to observe that, in a dataset, data belonging to certain labels have strong correlations with certain features that are not causally related to the identities of the data. For example, we can create a colored MNIST set where most 3s are blue and most 8s are red. Traditional statistics-based methods, including NNs, are likely to pick these correlated features as good indicators for the identities they correlate with. We say these features are spurious. Once the correlation is no longer there, for example in an out-of-distribution dataset, these methods will also misclassify data because of these learned spurious features. We hope that causality can help us avoid these spurious correlations through causal reasoning. We first investigate the traditional VAE and find that it is not able to conduct causal reasoning because the latent space is not semantic-consistent due to the regularization on the posterior of the latent variable. We propose a support-based regularization for the posteriors as a replacement to alleviate the negative effects the original regularization causes and a counterfactual generative goal to encourage a unified semantic space. Extensive experiments both on the simple colored MNIST and the more realistic CelebA dataset show that our new model is capable of causal reasoning, that is, generating counterfactual images. Furthermore, we propose to use our new framework to evaluate the probability of causation (POC) and use them to guide the learning of the representations, which rejects contributions made by spurious features. Representations acquired under POC are supposed to work on out-of-distribution samples, and our quantitative experiments show that our representation achieves much higher classification accuracy on out-of-distribution samples compared to other methods.

The rest of the chapter is organized as follows. First, we provide an intuitive introduction use the example of colored MNSIT to explain our motivation and ideas. Then we investigate current representation learning methods using causal concepts. After that, we formally introduce our generative model. We point out the issue with the traditional regularization term introduced in VAE and replace it with a relaxed alternative as well as a counterfactual task. Then we introduce the probability of causation and how we can use our VAE to evaluate them so that we can acquire causal representations. Finally, in experiments, we first show that our model is capable of creating counterfactual data required for our causal reasoning, and then we show that our model outperforms other methods in out-of-distribution classification tasks.

3.1 Introduction

Representation learning has always been an important task in the machine learning field. Real-world data such as images, audio, etc. are not convenient to directly process for downstream tasks like classification due to their nature of high dimensionality. The goal of representation learning is to acquire a mapping $f : \mathbf{x} \to \mathbf{r}$ from the data to a low dimensional vector, where $\mathbf{x}$ is the real-world data, and $\mathbf{r}$ is the representation, such that $\dim(\mathbf{x}) \gg \dim(\mathbf{r})$ while $\mathbf{r}$ is still able to preserve information about the characteristics of the individual data. The level of abstraction of these characteristics is also evolving. In the early days, people use representations, like SIFT [25], that are directly calculated from pixels. These representations are simple and stable, but are also low-level and require substantial effort to process, and they may lose useful information as well. As NNs are more and more popular, representations extracted from NNs gradually took over. The classic pipeline is as follows: First, we train a deep NN model on a dataset with the classification task. After the training is done, we extract a middle layer from the deep NN as the representation. You do not want to select layers too close to the image input, because then you would have a representation that is low-level and redundant, similar to the SIFT feature. On the other hand, you also do not want to select a layer too close to the output because by then, the network has abstracted away the majority of the information that is not directly helpful to the identification of these particular classes, and thus the generalizability of the representation would be pretty bad. Thus for an ideal deep

NN representation, it should preserve enough low-level information for generalization but also enough high-level abstraction to effectively compress the data without losing identity information. Unfortunately, due to the black-box nature of NNs, we can not really study how these representations are acquired during the training.

3.1.1 Spurious Features

A lot of the developments in representation learning are performance-oriented, we do not really have a mathematical theory of the definition of representation and what should count towards a good representation. A fundamental difficulty for the NN-based or, more generally, probability-based methods is that they rely on the assumption that all data are independently identically distributed (i.i.d.), meaning that these data are independently sampled from an identical distribution, while in real-world cases, we often face out-of-distribution (o.o.d.) data. For instance, camels usually appear on sand in images, and cows usually appear on grass. Because of the strong correlation between the animal and the background, the neural network trying to classify them might think the background is the key charactersitic of the animal. Now if we input an image of a camel on grass, the network might misclassify it as a cow.

In the following sections, we'll use colored MNIST as a running example to illustrate our ideas. Assume we have a colored MNIST dataset where number 3 is highly correlated with the color blue and number 8 is highly correlated with the color red. If we conduct the standard deep NN representation learning on this dataset and acquire a representation, blue may be in there (for example, as an entry of the representation vector), and considered as a good feature for the number 3, and red can also be considered as a good feature for the number 8. If we only use this representation on datasets with such correlation, this is going to work fine. But if we apply this representation where 3 and 8 are no longer correlated with the color or even reversely correlated (meaning 3 has a high correlation with red and 8 has a high correlation with blue), then the representation is going to underperform since the color is no longer an effective indicator of the number's identity. On the other hand, if we take this to the extreme where 3 is perfectly correlated with blue and 8 is perfectly correlated with red, meaning all 3s are blue and all 8s are red, then we consider these colors to be genuinely good representation

because we have no reason (from this dataset) to doubt 8 is ever going to be blue in an o.o.d. distribution since 8 has always been red, just like we will never doubt if the new 8 in a new dataset will have three circles because this has never happened. Of course, we know that color is not related to the identification of the number, it is the shape that actually matters. But we only know it because we have the oracle knowledge that machines do not possess. If we want our model to consistently judge which aspects should qualify as features, then we should also set rules that apply universally to all features, including colors and shapes. Similarly, if all number 7s has a dash in the middle, this should be considered a good feature for 7. If we encounter an o.o.d. 7 which does not have the dash, then it shouldn't be recognized as 7, even though we know what counts towards the identity of 7 and the new number is indeed a 7. In this thesis, we only consider a feature r spurious if it is supported on all marginal distributions conditioned on the labels, i.e. $P(R = r|l) > 0$ for $\forall l$.

3.1.2 Causal Representation

We resort to causal theory [36] to tackle the problem of o.o.d. data in representation learning and provide a theoretical framework. The power of causal models is that they provide semantics to directly modify the model and produce counterfactual data that do not exist or rarely appear in the original dataset. Assuming, the representation is a descendant of the label, the essential idea of using causality to evaluate how well a representation represents the identity of the data (labels) is to reason about the counterfactual question: Given the data (e.g., images), what would the representation be, had the label taken a different value (caused by our interventions). Such counterfactual reasoning comprises two facets, sufficiency and necessity. Sufficiency assesses the presence of a causal path through which an event can cause another event. On the other hand, necessity assesses the absence of alternative paths where an event does not reside, to cause another event. For example, in a fire accident, we have two events, A is the presence of oxygen, B is the result of a fire breakout. We consider A as a necessary cause because given the condition that a fire outbreak has happened and there is oxygen in place (factual); if there were no oxygen present (counterfactual), then the fire would not have happened. This means that even if there is an action to set fire, there is no

alternative mechanism to let the fire happen if there is no oxygen. But A is not a sufficient cause because given the condition that there is no fire outbreak and no oxygen, even if we had put oxygen in place, a fire would not break out since there is no action to directly cause the fire. On the other hand, assume we have two events in summer, A is buring a snow pile, and B is snow melting. A is considered a sufficient cause because given the condition that snow is still there and no one is burning it, had we burnt the snow, the snow would have melted because A is sufficient enough to single-handedly cause the melting. However, A is not a necessary cause because given the condition that we have burnt the snow pile and the snow has melted, even if we had not burnt it, the snow would still eventually melt since it is summer and the temperature is high enough to act as an alternative path to cause the melt.

By adopting these causal concepts into representation learning, we can rigorously reason about how a representation truly represents the identity. Take the colored MNIST as an example. Assuming we have a generative model, shown in Figure 3.1, that produces the colored MNIST dataset we have mentioned, where 3 is highly (but not perfectly, since we only consider features that happen to all labels spurious) correlated with the color blue, and 8 is highly correlated with the color red. Z_c is the context latent space controlling aspects other than identity, e.g., color, distortion, etc., and Z_l is the identity latent space that controls the class of the data, i.e., the number. The concepts of sufficiency and necessity are, by their nature, only compatible with binary (or more broadly categorical) events, but the output of a NN is required to be continuous because of gradient propagation. To fill this gap, we set up an additional mechanism that samples binary representation values $r_i \in \{0, 1\}$ from distribution Bernoulli(v_i) parameterized by a continuous variable v_i captured by the NN f. Now if we are given an image of blue 3 and we have a trained NN f to capture the representation where color is one of the entries. We want to test out if the color blue is a good representation that can consistently indicate whether the identity of the number is 3. Let's revisit and evaluate the counterfactual question that we proposed: "Given the image, had the label Z_l taken a different value, what would the representation be?". First, we can infer Z_c and $Z_l = 3$ from the image, where the color as well as other context information is within Z_c. Then we intervene on Z_l and change it to 8, but preserve the Z_c inferred from the original blue 3, then ideally we can produce a counterfactual image of blue 8 which is quite rare in the dataset, considering 8s

are mostly correlated with the color red. This is the counterfactual image of "what would the image be, had the label Z_l taken a different value?". Then the counterfactual image is fed into the representation function f to produce V. Eventually, we can sample the representation R from V via Bernoulli. Now if we revisit r that represents the color, we would find that r did not change since the color did not change after the intervention, meaning that the probability of necessity of the image being a 3 to the image being blue is low, indicating that blue is not a sufficient cause for the number 3 (if A is necessary for B, then B is sufficient for A). Hence we are able to tell from counterfactual experiments that the color is a spurious feature for identifying the numbers. In conclusion, if we want to find a representation that can causally determine the identity of the image, then we should look for representations that have a high probability of sufficiency and necessity to identify the label of the image.

In addition, the probability of causation is defined for two binary events. Since each representation may be causally useful for multiple labels, it is infeasible to calculate POC for each representation and all labels respectively. Instead, we still frame f as a classification network where we have L output nodes, each representing the score for a category. We calculate POC for these output entries and their corresponding labels so that if a spurious feature contributed to the final prediction, the gradient propagation will be able to eliminate that feature or the path through which the feature contributes to the final results. We also add a norm-based regularization term on the parameters of the model so that if f accidentally picks up a spurious feature and POC stops it from being used, the regularization term will be able to get rid of it.

3.1.3 Counterfactual VAE with a Unified Semantic Latent Space

However, the above causal reasoning relies on the premise that our generative model is indeed capable of producing correct counterfactual images, that is, intervening the label on a blue 3 produces a blue 8 instead of a red 8. We use the variational autoencoder (VAE) to model the probabilistic process. However, VAE imposes a regularization on the posterior of $P(Z_c|Z_l)$ that it should stay close to a standard multivariable Gaussian $\mathcal{N}(0, 1)$. The purpose of this regularization is to force all Z_cs inferred from the real data given a specific label Z_l to stay close to the standard Gaussian distribution so that when we sample a latent variable from

the Gaussian and use it to generate a synthetic image, it would look realistic. However, it contradicts our goal of counterfactual generation. Consider the colored MNIST example, assume R is the variable controlling the color. If the VAE is trained well and the regularization on the posterior is well fulfilled, that is, $P(r|Z_l = 3) = \mathcal{N}(0, 1)$ and $P(r|Z_l = 8) = \mathcal{N}(0, 1)$. We know the actual color distributions given different labels are different. For example, most of the random noise sampled from the Gaussian will be translated to blue when $Z_l = 3$ (because there is a strong correlation between blue and 3), and the same set of noise will be translated to mostly red when $Z_l = 8$. As a result, a variable sampled from $\mathcal{N}(0, 1)$ may be translated to different colors when different labels are given. If we intervene the label on a blue 3 to $Z_l = 8$, instead of producing a blue 8, it will most likely produce a red 8, which undermines the purpose of counterfactual reasoning. What we actually want is not $Z_c \perp Z_l$, on the contrary, we want Z_c to be dependent on Z_l so that Z_c can encode consistent semantic information across different labels, that is, the same r that represents blue for $Z_l = 3$ is also going to represent blue when $Z_l = 8$. We call such a latent space where variables within it have the same semantic meaning across different labels unified semantic space. To grant the VAE model with the ability of counterfactual generation, we relaxed the regularization on the posterior and add a counterfactual task where for each input data, we conduct an intervention to change the label to a counterfactual label and produce the counterfactual image with it. Then we feed the counterfactual image back to the recognition model in VAE to teach the model to correctly classify the new counterfactual image to the counterfactual label, which will further encourage the model to learn a unified semantic latent space.

Contributions: 1) We point out that regular VAE models are not able to conduct counterfactual reasoning because of the regularization imposed on the posterior of the latent variable. We further propose to relax the regularization and use a counterfactual generation task to learn a unified semantic space and enable the model to conduct correct counterfactual reasoning. 2) We use our novel VAE to evaluate the probability of causation and use it to guide the representation learning, which will reject contributions from spurious features and thus exhibit resistance to out-of-distribution data.

3.2 Background

3.2.1 Motivation

Ever since the introduction of representation learning, people have been trying to achieve "better" features that contain critical information. The classification accuracy of a representation has been the main measure. However, such a measure is based on statistics and thus can not expose problems related to causality. For instance, in the colored MNIST example, assume 95% of 3s are blue and 95% of 8s are red, if in the original grayscale MNIST dataset, our model can achieve 90% accuracy based on the actual features (shapes) of these numbers, then in the colored MNIST case, the model can easily achieve a 95% accuracy by simply using the color as the representation. Surely the rarely colored red 3s and blue 8s are misclassified, but from the perspective of statistics, it is perfectly fine to sacrifice these samples in order to achieve higher accuracy. However, once we switched to a colored MNIST dataset that has the opposite correlation, e.g., 95% of 3s are red and 95% of 8s are blue, then the model that considers color a good feature would only get 5% of the data correctly classified. We refer to these data with a different distribution as out-of-distribution data. We wish to introduce causality into our current approach and methodology so that we can learn representations less prone to these problems caused by distribution change. Current causal representation learning methods can be mainly divided into two branches. First is the single-environment method where we only assume access to a single dataset. Generally, it is impossible to get the true causal relationships from a single distribution. However, by giving additional assumptions and rules, we can define what accounts for a good representation. Second is the multi-environment method where we assume access to multiple datasets. By utilizing the differences between these datasets, we are going to be able to expose the causal relationships. When sufficient different datasets are given, the causal structure will eventually become identifiable.

3.2.2 Multi-Environment methods

For these methods, we consider the setting where multiple datasets $\mathcal{D}_e = \{X_e^{(i)}, Y_e^{(i)}\}_{i=1}^{N_e}$, $e \in \mathcal{E}$, are available. Peters et al. [38] proposes invariant causal prediction (ICP) which explores the causal interpretation of invariance to find the direct causes of a variable of interest. More specifically, consider the model $Y_e = \mu + \gamma^T \mathbf{X}_e + \epsilon_e$, where $\mathbf{X}$ is the representation, Y is the prediction target, ϵ is the noise which is unique for each individual environment. If we can find a set of parameters μ, γ such that this relationship holds true in all environments, then these representations are considered the true causes. Note that we assume there are no other variables involved and thus there should be no confounders for these representations. The issue with ICP is that it is very difficult to construct a causal graph to integrate all the observable variables into a single model, and the assumption that there really is a complete structural causal model relating these variables is also too strong for realistic cases. Arjovsky et al. [1] propose *invariant risk minimization* (IRM) to learn direct causes of target variable without learning the true underlying causal graph. Formally, a representation $\Phi : \mathcal{X} \to \mathcal{H}$ is said to elicit an invariant predictor $w : \mathcal{H} \to \mathcal{Y}$ if this predictor w is optimal across all environments. It seems to be a bi-level optimization problem where we need to find the representation function Φ and the classifier w simultaneously, however, the authors claim a naive classifier $w|w = 1.0$ is good enough and we can leave the actual job of classification to the representation algorithm Φ. Eventually, the optimization goal is

$$\min_{\Phi:\mathcal{X}\to\mathcal{Y}} \sum_{e\in\mathcal{E}} R_e(\Phi) + \lambda \cdot ||\nabla_{w|w=1.0} R_e(w \cdot \Phi)||^2. \tag{3.1}$$

The first term R_e is the risk under environment e. The gradient norm term is to measure the optimality of the naive classifier for the representation Φ under environment e. If the overall classifier $1 \cdot \Phi(\mathbf{X})$ for $\mathbf{X}$ is indeed optimal, then the gradient propagated to w should be zero. For a single environment, of course, you can find many optimal solutions, just like we do not have identifiability for a NN classifier. This problem will not be solved by adding one or a few more environments/datasets because you would still have a lot of potential spurious structures inside the model. The authors prove that when enough different environments are given, eventually we are going to be able to exhaust all alternative causal structures, and we will find

such a representation that can achieve invariant optimal prediction performance across all environments. Intuitively speaking, different environments act as restrictions, and satisfying more of these restrictions reduces the degree of freedom, when the degree of freedom is reduced to zero, we will be able to get a unique solution. However, such a prerequisite is too strong to practically take advantage of it. In fact, Rosenfeld et al. [43] proves that for linear cases, only when the number of environments is larger than the dimensionality of the environmental features, that is, randomness that corresponds to each individual environment and varies across different environments, can we use IRM to find the predictor. And for non-linear cases, IRM would fail catastrophically unless the test data are sufficiently similar to the training distribution and thus IRM would not achieve better results on non-linear cases than regular empirical risk minimization (ERM). Further, to extend from linear cases to non-linear cases, Lu et al. [26] proposes invariant Causal Representation Learning (iCaRL) to model the data with an identifiable variation autoencoder based on iVAE and use IRM to guide the training.

In addition, if we know the process well enough and can design a causal graph that's close enough to reality, then we can also identify the causal relationships using interventions and produce causal representation from the causal graph [56]. However, these causal graphs are usually too complicated to design for vision tasks.

Overall, these multi-environment methods, although more theoretically grounded, are also very limited by the multi-environment assumptions. For image datasets, the multi-dataset setting is especially hard to set up, because, more often than not, we will not have multiple datasets from different environments to decrease the degree of freedom of the model all the way down to zero to achieve identifiability.

3.2.3 Single-Environment methods

Assuming that we only have access to a single dataset is a much more realistic setting, because, in most scenarios, especially for computer vision datasets, we only have one dataset, and we don't have the apparatus or the resources to construct another dataset with different underlying

causal processes (by, e.g., retaking pictures with a different approach that will form a new distribution of images). Assuming X is the data, Y is the label, and Z is the latent variable that controls other aspects, like background, we refer to Z as context information. Makar et al. [27] and Puli et al. [41] propose to reweight the data to break dependence between Z and Y so that all spurious correlations can be eliminated, which is not easy in practical situations. Veitch et al. [54] propose counterfactual invariance which states that $f(X|Z) = f(X|Z')a.e.$. It demands a representation that is invariant under different context variables, which we believe is too strong. In many situations, especially for images, representation is bound to preserve much information that is useless under the current setting but proved useful when the distribution changes, which is part of the reason we want to learn representation in the first place. It also requires Z to be discrete, which is too restrictive for computer vision tasks. Wang and Jordan [57] propose to also utilize probability of causation to guide the learning of representations. Because they do not have a generative model, the counterfactual terms of POC are consequently unidentifiable. They use relaxation to turn the counterfactual term into an interventional lower bound, which may lose a lot of information. These previous works have focused on the independence relationships between X, Y, and Z, we argue that statistical independence is not even a plausible property we need for counterfactual. On the contrary, we can let the model preserve the dependence and save us the trouble of building the correct causal graph when conducting counterfactual generation. Additionally, almost all of these works are verified only on simple datasets like synthetic few variable models.

3.2.4 Structural Causal Model with NNs

There have been some previous works that try to combine SCMs with VAE. [60] proposed to use an SCM to model the relationships between the latent variables in a VAE. However, they did not use any counterfactual or interventional goal to guide the learning of the model. It is well understood that you can not identify causal directions/relationships between variables with only observational statistical information. Note that by observational statistical data, we mean a singular distribution. If you have datasets of different distributions (environments), then they can be considered as distributions under different interventions and thus are no

longer observational. They claimed that their model is capable of generating counterfactual images by intervening in the latent variables. This is not unexpected because if the data is balanced and there is no strong correlation between context variables, you may get a latent space with consistent semantic meanings, and the counterfactual intervention done on the data will also make sense. But as we will show, once there is a strong correlation, it is impossible to get a unified semantic space without leveraging the power of counterfactual goals.

We aim to propose a method that works in the single-environment setting, with a modified VAE capable of producing meaningful counterfactual images and we use unrelaxed POC to guide the learning of causal representation.

3.2.5 Probability of Causation

People have long realized that cause-effect has two distinct facets, *sufficiency* and *necessity* [36], which should be considered together when constructing causal explanations. Assuming two events A and B are binary, i.e., $A, B \in \{0, 1\}$. We have *probability of necessity* (PN) to assess how necessary A is for B to happen, defined as

DEFINITION 3.2.1 (Probability of Necessity, PN [36]). *Let $A = 1$ and $B = 1$ indicate that A and B happen or turn on. The probability of necessity is defined as*

$$\mathrm{PN} = P(B_{A=0} = 0 | A = 1, B = 1). \tag{3.2}$$

This reads: given A turned on ($A = 1$), and B turned on, what is the possibility of B turning off if we intervene and force A to turn off (do$(A = 0)$)? Note that the necessity of A is measured against the reality where A and B are both turned on, making sure that there are no other factors that can solely prevent B right now. As a result, if we turn off A, and B is consequently turned off, then we can say this is all A's doing, and A is a necessary cause for B. Here $B_{A=0}$ refers to a B in an alternative reality where A did not happen, hence we describe such variables as counterfactual. Similarly, we also have *probability of sufficiency* (PS) of A to B as $\mathrm{PS} = P(B_{A=1} = 1 | A = 0, B = 0)$. These probabilities based on counterfactual variables are called probabilities of causation (POCs). By combining

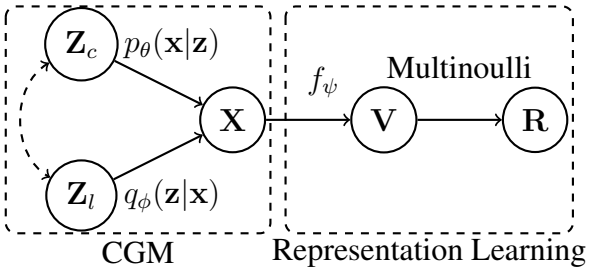


FIGURE 3.1. **Structural Causal Model.** We construct our causal representation learning framework as shown in the paradigm. Take the colored MNIST as an example, $\mathbf{Z}_c$ encodes all identity irrelevant information, such as color, distortion, etc. $\mathbf{Z}_l$ represents the identity label, i.e., which number is in the image. Such separation allows us to conduct causal reasoning to find what information can truly determine the identity of the image and exclude those that only have spurious correlations with the label. While $\mathbf{Z}_c$ and $\mathbf{Z}_l$ may be confounded, we develop a theory that enables us to conduct meaningful counterfactual manipulation without modeling the confounders. $\mathbf{X}$ is the image, f is a functional mapping that can be trained to only extract causal information from $\mathbf{X}$ and produce an intermediate continuous label prediction results $\mathbf{V}_o$. Finally, $\mathbf{V}_o$ goes through a Bernoulli process to get the binary label prediction results.

these two notions, we also have *probability of necessity and sufficiency* (PNS) defined as $\mathrm{PNS} = P(B_{A=1} = 1, B_{A=0} = 0) = P(A = 0, B = 0)\mathrm{PS} + P(A = 1, B = 1)\mathrm{PN}$.

3.3 Counterfactual Variational Autoencoder

3.3.1 Structural Causal Model

As we have mentioned in Section 3.1.2, in order to evaluate the causal explanatory power of a representation over the final predicted label, we need to construct a generative model that is capable of producing counterfactual data. Since we want to study the causal relationship between the label and the representation and conduct potential interventions and supervision on the label, it is necessary to disentangle the overall latent space into the context latent space and the label/identity latent space. We model the whole process as Figure 3.1. Latent space $\mathbf{Z}$ is disentangled into a latent context variable $\mathbf{Z}_c$ and a latent identity variable $\mathbf{Z}_l$ ($\mathbf{Z}_l$ is essentially a categorical label, which is usually represented by a one-hot vector), which may be

confounded by unobserved factors. $p_\theta(\mathbf{x}|\mathbf{z})$ is the generative distribution of high dimensional data $\mathbf{x}$ given the latent space $\mathbf{z}$, parameterized by θ. We also have a recognition model $q_\phi(\mathbf{z}|\mathbf{x})$, parameterized by ϕ, to approximate the posterior of $\mathbf{z}$ given $\mathbf{x}$. On the representation learning side, we have f_ψ to gain the underlying representation $\mathbf{V} = f(\mathbf{X})$ from $\mathbf{X}$, and the real representation $\mathbf{R}$ is sampled from a series of Bernoulli distributions parameterized by $\mathbf{V}$, i.e., $R_i \sim \text{Bernoulli}(V_i)$ where $V_i \in (0, 1)$ is a real number. The reason we set up this extra layer of mapping from $\mathbf{V}$ to $\mathbf{R}$ is that the probability of causation that we are going to use to evaluate causal effects is only compatible with binary events or categorical events at best, thus we need a probability process to fill the gap between the continuous output variables produced by the NN p_θ and the binary representation required by the probability of causation.

In traditional representation learning with NNs, people usually select an intermediate hidden layer in the NN as the representation, which, in equivalence, would be a layer of the NN in p_θ or a layer of q_ϕ. Why didn't we simply do the same and select a layer from the VAE as the representation? From the perspective of causal models, representation learning is about only selecting variables/features that can causally determine or help to determine the label, some information critical to the generative model may not be important for our causal representation learning. For example, in the case of colored MNIST, for any layer in p_θ or q_ϕ, you need to have at least a variable that encodes the information about the color. If p_θ does not have it, then the model is not able to generate images of different colors, similarly, if q_θ does not have it, then the recognition model would fail to decode the information about the color into $\mathbf{Z}_c$ and consequently fail to reconstruct an image of the same identity and color in the reconstruction process. However, as we have discussed, in situations where numbers have correlations with colors, color is not a good representation which may lead to misclassification when applied to o.o.d. samples with a very different correlation between labels and colors. Thus we set up an extraneous function f_ψ that extracts causally stable information (rid of spurious correlations) from the high dimensional data $\mathbf{X}$ so that we do not need to worry about discarding information that is harmful to label prediction but crucial for the training of the model, especially for the reconstruction part.

3.3.2 Variational Inference

We can use a generative model to describe the distribution of images, which is basically a probabilistic distribution. Variational inference is a method to approximate the posterior of the latent variable given the data, so that we can potentially describe the underlying latent space for the image dataset.

The joint distribution for the complete generative model is

$$p(\mathbf{Z}, \mathbf{X}, \mathbf{V}, \mathbf{R}) = p(\mathbf{Z}_c, \mathbf{Z}_l)p(\mathbf{X}|\mathbf{Z}_c, \mathbf{Z}_l)\delta(\mathbf{V} - f(\mathbf{X}))P(\mathbf{R}|\mathbf{V}), \tag{3.3}$$

where $\delta(\cdot)$ is a Dirac delta function. The only variables that are observable to us are $\mathbf{X}$ and $\mathbf{Z}_l$ which are the high dimensional data and the identity label respectively. The dataset is $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{z}_l^{(i)})\}_{i=1}^N$, the marginal likelihood for the two variables X and Z_l is

$$\log p(\mathbf{x}^{(1)}, \mathbf{z}_l^{(1)}, \ldots, \mathbf{x}^{(N)}, \mathbf{z}_l^{(N)}) = \sum_i \log p(\mathbf{x}^{(i)}, \mathbf{z}_l^{(i)}), \tag{3.4}$$

which is the sum of the individual marginal likelihood of the data. From now on, we focus on the individual marginal likelihood $\log p(\mathbf{x}^{(i)}, \mathbf{z}_l^{(i)})$. For simplicity, we also remove the superscript for now and use $\mathbf{x}$ and $\mathbf{z}_l$ to denote the data and the label for an individual piece of data. By integrating over a distribution $q(\mathbf{z})$ on $\mathbf{Z}$, we have the following relaxation on the likelihood:

$$\log p(\mathbf{x}, \mathbf{z}_l) = \int q(\mathbf{z}) \log p(\mathbf{x}|\mathbf{z}_l) d\mathbf{z} + \log p(\mathbf{z}_l) \tag{3.5}$$

$$= \int q(\mathbf{z}) \log \frac{p(\mathbf{x}, \mathbf{z}|\mathbf{z}_l)}{p(\mathbf{z}|\mathbf{x}, \mathbf{z}_l)} d\mathbf{z} + \log p(\mathbf{z}_l) \tag{3.6}$$

$$= \int q(\mathbf{z}_c) \log \frac{p(\mathbf{x}, \mathbf{z}_c|\mathbf{z}_l)}{q(\mathbf{z}_c)} + q(\mathbf{z}_c) \log \frac{q(\mathbf{z}_c)}{p(\mathbf{z}_c|\mathbf{x}, \mathbf{z}_l)} d\mathbf{z}_c + \log p(\mathbf{z}_l) \tag{3.7}$$

$$= \int q(\mathbf{z}_c) \log \frac{p(\mathbf{x}, \mathbf{z}_c|\mathbf{z}_l)}{q(\mathbf{z}_c)} d\mathbf{z}_c + \log p(\mathbf{z}_l|\mathbf{x}) + D_{KL}(q(\mathbf{z})|p(\mathbf{z}|\mathbf{x})) \tag{3.8}$$

$$\geq \int q(\mathbf{z}_c) \log \frac{p(\mathbf{x}, \mathbf{z}_c|\mathbf{z}_l)}{q(\mathbf{z}_c)} d\mathbf{z} + \log p(\mathbf{z}_l). \tag{3.9}$$

We usually use

$$\mathcal{L}(\mathbf{x}) \coloneqq \int q(\mathbf{z}) \log \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} d\mathbf{z} \tag{3.10}$$

to denote the first term as the Evidence Lower Bound (*ELBO*). The second term is the loss term for the discriminative task of inferring $\mathbf{z}_l$ from $\mathbf{x}$. When $\mathbf{Z}_l$ is categorical, $\log p(\mathbf{z}_l|\mathbf{x})$ is a cross-entropy loss. Since $q(\mathbf{z}) = q(\mathbf{z}_c, \mathbf{z}_l)$, $\mathbf{z}_l$ is given as part of the data, integrating over $q(\mathbf{z})$ is equivalent to integrating over $\mathbf{z}_c$. Also because the only requirement for $q(\mathbf{z})$ in Equation 3.10 is that it is a probability distribution, which means that $q(\mathbf{z})$ can be conditional on $\mathbf{x}$ or $\mathbf{z}_l$, we can substitute $q(\mathbf{z})$ with $q(\mathbf{z}_c|\mathbf{x}, \mathbf{z}_l)$, and the ELBO can be rewritten as

$$\mathcal{L}(\mathbf{x}) = \int q(\mathbf{z}_c) \log \frac{p(\mathbf{x}, \mathbf{z}_c|\mathbf{z}_l)}{q(\mathbf{z}_c)} d\mathbf{z}_c \tag{3.11}$$

$$= \int q(\mathbf{z}_c|\mathbf{x}, \mathbf{z}_l) \log \frac{p(\mathbf{x}, \mathbf{z}_c|\mathbf{z}_l)p(\mathbf{z}_c|\mathbf{z}_l)}{q(\mathbf{z}_c|\mathbf{x}, \mathbf{z}_l)p(\mathbf{z}_c|\mathbf{z}_l)} d\mathbf{z}_c \tag{3.12}$$

$$= \int q(\mathbf{z}_c|\mathbf{x}, \mathbf{z}_l) \log \frac{p(\mathbf{x}, \mathbf{z}_c, \mathbf{z}_l)p(\mathbf{z}_l)}{p(\mathbf{z}_c, \mathbf{z}_l)p(\mathbf{z}_l)} \frac{p(\mathbf{z}_c|\mathbf{z}_l)}{q(\mathbf{z}_c|\mathbf{x}, \mathbf{z}_l)} d\mathbf{z}_c \tag{3.13}$$

$$= \int q(\mathbf{z}_c|\mathbf{x}, \mathbf{z}_l) \log p(\mathbf{x}|\mathbf{z}_c, \mathbf{z}_l) d\mathbf{z}_c - D_{KL}(q(\mathbf{z}_c|\mathbf{x}, \mathbf{z}_l)|p(\mathbf{z}_c|\mathbf{z}_l)). \tag{3.14}$$

The first term is a reconstruction loss, where the model infers the distribution of $\mathbf{z}_c$ given $\mathbf{x}$ and $\mathbf{z}_l$, and then uses the given condition $\mathbf{z}_l$ and the posterior of $\mathbf{z}_c$ to reconstruct the high-dimensional data and see how close the reconstructed data is to the original input $\mathbf{x}$ through a log-likelihood function. The second term, on the other hand, is a regularization term that encourages the posterior $q(\mathbf{z}_c|\mathbf{x}, \mathbf{z}_l)$ to be like the prior $p(\mathbf{z}_c|\mathbf{z}_l)$ as much as possible through a KL divergence, which equals 0 when the two distributions are identical. By fulfilling the posterior regularization term, the model can restrict the latent variables inferred from different data in a tight area around the prior, and consequently, enhance the generative power of the model. The prior $p(\mathbf{z}_c|\mathbf{z}_l)$ is usually set as a standard noise distribution like standard Gaussian for VAE. When you want to generate synthetic samples from the model, you would want your posterior $\mathbf{z}_c|\mathbf{z}_l$ to be as close to the prior as possible so that synthetic images generated by sampling $\mathbf{z}_c$ from the Gaussian would also look similar to real images. So far, this is a weakly supervised variational Bayes model with labels for us to supervise on.

3.3.3 Inconsistent Semantics for the Latent Space

Under this generative framework, a piece of data is encoded into two parts, Z_c which controls the aspects unrelated to identity, e.g., color, distortion etc., and Z_l which is the label or identity of the data. To conduct counterfactual image generation, we can preserve Z_c and change Z_l to generate new data that does not comply with the current dataset distribution. For instance, in the colored MNIST dataset, let's say the number 3 is highly correlated with the color blue, and the number 8 is highly correlated with the color red, if you want to consistently generate red 3, then you can use $q(\mathbf{z}|\mathbf{x})$ to encode red 8 into z_c and $z_l = 8$[1], and then change Z_l to 3, and conduct the inference $p(\mathbf{x}|\mathbf{z})$ to generate a new image $\mathbf{x}_{Z_l=3}$. The variable controlling the color to red is still in Z_c. If Z_c does its job and preserves the actual color, then the new number is still going to be red, also because the identity has been changed to 3, the generated image $\mathbf{x}_{Z_l=3}$ is going to be a red 3 will is very rare in the original dataset. This is how we leverage the power of causal models to conduct counterfactual reasoning, we are basically fabricating data that does not exist or rarely exists in the original distribution.

However, such counterfactual generation relies on the premise that Z_c stores values that have the same semantic meaning across all labels. The graphical structure of the SCM guarantees that the variable that directly controls the color is in Z_c because, take the colored MNIST as an example, given the same Z_l, images are still going to have all the colors, which means that a part of $Z_c \in \text{supp}(p(Z_c))$ maps to blue numbers and the rest maps to red numbers. This is true for both 3 and 8. However, since the two numbers have different color distributions, there may exist a mechanism that can generate conditional color distributions, which means that the Z_c that generates blue 3 may generate red 8 when we intervene on the label. This will invalidate our counterfactual goal because we want to only modify the label while leaving the rest of the features, including color, untouched. We need to ensure that variables in Z_c have the same semantic meaning whichever Z_l is given.

Next, we show that under the standard variational autoencoder assumptions, the NN is bound to create internal mechanisms to generate conditional distributions for different labels.

[1] As we have mentioned, $\mathbf{Z}_l$ should be a one-hot vector representing the identity label. For simplicity, we also use $Z_l = 3$ or $Z_l = 8$ to indicate that the label is 3 or 8 when there is no ambiguity.

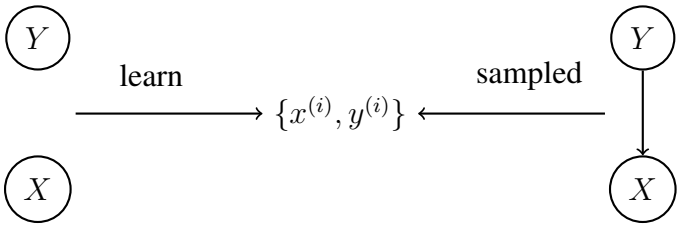
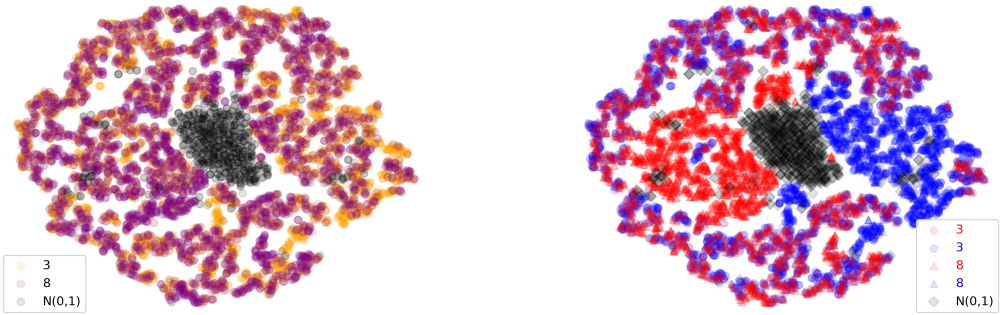


FIGURE 3.2. Model with independent prior modeling data sampled from dependent joint distribution



(A) $p(\mathbf{Z}_c|Z_l)$ (B) $p(\mathbf{Z}_c|Z_l, \text{color})$

FIGURE 3.3. **tSNE embedding for a vanilla VAE.** A vanilla VAE is trained on the colored MNIST dataset, and all $\mathbf{z}_c$ are trained under one tSNE embedding. 3.3a shows the distribution of $\mathbf{Z}_c$ conditioned on Z_l. The different colors of nodes represent different identities. 3.3b shows the distribution of $\mathbf{Z}_c$ conditioned on color and Z_l. The colors of nodes represent the actual color of the image, while the identities of images are represented by the shape of nodes. From the embedding shown in 3.3a, we can see that the posterior distributions of $\mathbf{Z}_c$ given whichever Z_l is basically the same as they have the same support and the same densities, i.e., $\mathbf{Z}_c \perp Z_l$, which is intended by VAE but at the same time bad for counterfactual image generation. From the embedding shown in 3.3b, we can see that different numbers of different colors are overlapped in the $\mathbf{Z}_c$ space, meaning the counterfactual images generated by intervening the label may end up with different colors, which would demise the purpose of counterfactual reasoning, which is to generate counterfactual images based on real data with different identities while preserving all identity-unrelated traits, including the color.

Furthermore, if those assumptions are well met, Z_cs inferred from images with different Z_ls are guaranteed to have inconsistent semantic meanings. A common assumption that we impose on VAE is that the prior $p(\mathbf{z}_c|\mathbf{z}_l)$ (as opposed to the posterior $q(\mathbf{z}_c|\mathbf{x}, \mathbf{z}_l)$) is a standard

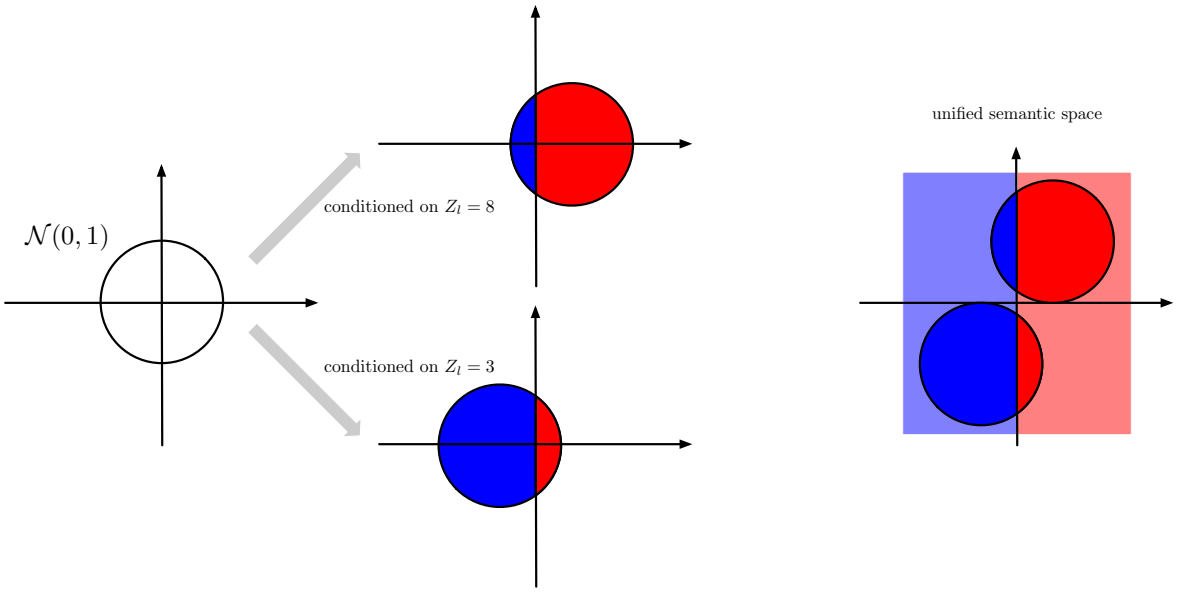


FIGURE 3.4. **Conditional color distribution vs unified color distribution.** Having the posterior $p(\mathbf{z}_c|\mathbf{z}_l)$ conforming to the standard Gaussian $\mathcal{N}(0,1)$ and having a unified semantic space for $\mathbf{z}_c$ are essentially two mutually exclusive things. Assume Z_c is a univariate variable directly responsible for producing images of different colors for colored MNIST, the conditional distributions of color are different given different labels Z_l. For instance, given $Z_l = 3$, the color is more likely to be blue. However, being the variable that directly controls color does not mean that it will produce an image of the same color when a different label Z_l is given. A z_c that produces a blue 3 may produce a red 8 when $Z_l = 8$ is given. If the VAE model is perfectly fitted, then $p(Z_c|Z_l = 3)$ and $p(Z_c|Z_l = 8)$ both conform to $\mathcal{N}(0,1)$, the model is bound to learn a mechanism inside the NN to produce the different conditional distributions of color, as shown on the left-hand side. However, if we are able to guide the model to learn a unified semantic space for Z_c, meaning that Z_c under different Z_l encodes the same semantic information, then the same Z_c that produces blue images is also going to produce blue images when a different Z_l is given.

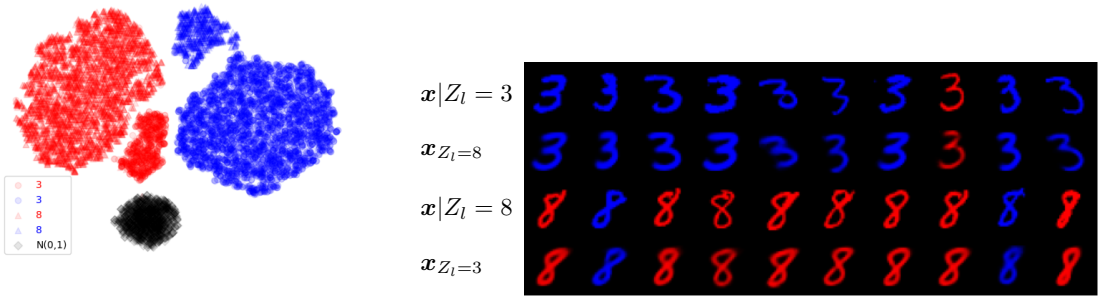
noise distribution, e.g., a Gaussian. In the colored MNIST example shown in Figure 3.4, let's assume Z_i is the variable controlling the color and the posterior of Z_i given Z_l is a standard noise distribution. For simplicity, we assume $Z_i|Z_l \sim \text{Uniform}(0,1)$. Then you would need a conditional functional mechanism later in the NN $p_\theta(\mathbf{x}|\mathbf{z})$ that can map this variable Z_i to the

actual color $f_c(\cdot|Z_l) : (0, 1) \mapsto \{\text{red}, \text{blue}\}$. An ideal set of mechanisms would be

$$f_c(Z_i|Z_l = 3) = \begin{cases} \text{red} & \text{if } Z_i \geq 0.9 \\ \text{blue} & \text{o.w.} \end{cases}, f_c(Z_i|Z_l = 8) = \begin{cases} \text{red} & \text{if } Z_i \geq 0.1 \\ \text{blue} & \text{o.w.} \end{cases}, \quad (3.15)$$

where generated images of 3 are mostly (90%) blue and images of 8 are mostly red. The purpose of restricting the posteriors to a standard noise distribution is that when you want to generate new images via sampling from the noise distribution, the sampled latent variable will be close to latent variables inferred directly from real data. However, this regularization will also force the latent variable to possess different semantic meanings under different labels. For $0.1 \leq Z_i < 0.9$, Z_i represents blue when $Z_l = 3$ and represents red when $Z_l = 8$. In other words, 80% of all images would have a different color had the label been different, which completely fails our counterfactual reasoning. When we check if the color blue is a good feature for 3, essentially we want to create an "imaginary" image where features that are not firmly bonded with the identity are preserved, but the actual number has been changed to, e.g., 8. If the color changes during this procedure, then it confirms the theory that the color is part of the identity of the number. As we have shown, 80% of all images will endorse this spurious theory because they will have a different color after the intervention. As a result, a regular VAE trained under the setting where strong spurious correlations are present is going to consider the spurious feature as a good representation of the identity that they correlate with. The experiment shown in Figure 3.3 confirms our theory. As we can see, the vanilla CVAE constraint the prior $p(\mathbf{z}_c|\mathbf{z}_l)$ to the same standard Gaussian distribution. Thanks to the regularization, we have $\mathbf{z}_c \perp Z_l$ since $p(\mathbf{z}_c|Z_l = 3) = p(\mathbf{z}_c|Z_l = 8)$ meaning that they have the same densities for any $\mathbf{z}_c$ in the area, shown in Figure 3.3a. As we have discussed, this forces the latent variable to have different semantic meanings under different labels, shown in Figure 3.3b, images in the area where blue and red dots are mixed are the ones that will not preserve color after the intervention to change the label.

The traditional regularization on the posterior is definitely working against our intention to create a consistent semantic latent space. So can we relax the regularization to alleviate this? Simply removing the regularization is certainly not going to work, as shown in Figure 3.5. Without the regularization to hold them together, the conditional posteriors $p(\mathbf{z}_c|\mathbf{z}_l)$ would



(A) tSNE for Z_c when prior is relaxed. (B) Real vs. counterfactual images when Z_l is intervened.

FIGURE 3.5. **Vanilla VAE with relaxed posterior.** As shown in Figure 3.5a, by relaxing the posterior regularization imposed on $p(\mathbf{z}_c|\mathbf{z}_l)$, we successfully separate the image dataset into different semantic spaces, meaning that the same $\mathbf{Z}_c$ that produces a blue image will not produce a red image when the label Z_l is intervened. However, only relaxing the posterior is far from enough and produces a severe problem. $\mathbf{z}_c$s that are inferred from images of 3 do not have the experience of producing images of 8 and vice versa, which means that intervening on Z_l will be an out-of-distribution behaviour and may produce unexpected results. As shown in Figure 3.5b, the counterfactual images generated by changing the label still have the same identities, because NNs are never trained to handle the new latent space ($\mathbf{Z}_c|\{Z_l = 3\} \times \{Z_l = 8\}$ or $\mathbf{Z}_c|\{Z_l = 8\} \times \{Z_l = 3\}$, $\times$ is the direct product).

be away from each, shown in Figure 3.5a, and since the generative mechanism for e.g., the number 3 has never encountered latent variables inferred from 8s, the counterfactual images generated by intervention would produce expected results, shown in Figure 3.5b. If Z_c truly represents the semantic meaning, that is, there exists a mapping f_s that applies to all images regardless of the labels they belong to, such that f_s maps Z_c to the true semantic meanings. Images belonging to different labels are going to have arbitrarily different distributions over the domain of Z_c and we should not let this information influence the training of our model, which means that our regularization should not be related to the densities of the posterior $p(\mathbf{z}_c|\mathbf{z}_l)$. Based on this idea, we propose a support-based regularization on the posterior of $\mathbf{Z}_c$ as

DEFINITION 3.3.1 (Posterior Support Difference). *Let $Z_l \in \{Z_l^1, Z_l^2, \dots Z_l^L\}$ be all labels, $p(\mathbf{z}_c|Z_l^i)$ is the posterior of $\mathbf{z}_c$ given the particular label Z_l^i. The common support is defined*

as

$$\textit{supp}_c = \bigcap_i \textit{supp}(p(\mathbf{z}_c|Z_l^i)). \tag{3.16}$$

The posterior support difference is defined as

$$M = \sum_i m(\textit{supp}(p(\mathbf{z}_c|Z_l^i))\backslash \textit{supp}_c), \tag{3.17}$$

where $m(\cdot)$ *is a measure,* $A\backslash B = \{x|x \in A, x \notin B\}$.

Because a lot of distributions we use, like Gaussian, have a support that fills the whole space, we can relax the definition of support to

DEFINITION 3.3.2 (Relaxed Support). *let* $p(\mathbf{x})$ *be a distribution of* $\mathbf{x}$*, which is Lebesgue integrable. For a real number* $0 < \epsilon \leq 1$*, there exists* d *such that*

$$\epsilon = \int_{p(\mathbf{x})>d} p(\mathbf{x})d\mathbf{x} \tag{3.18}$$

The relaxed support of $\mathbf{x}$ *is defined as*

$$\widehat{\textit{supp}}(\mathbf{x}) = \{\mathbf{x}|p(\mathbf{x}) > d\}, \tag{3.19}$$

where ϵ *controls how relaxed the support should be. When* $\epsilon = 1$ *(and consequently* $d = 0$*),* $\widehat{\textit{supp}}(\mathbf{x}) = \textit{supp}(\mathbf{x})$.

By combining these two concepts, the ideal maximal regularization we want to impose on the posterior would be

$$\hat{M} = \sum_i m(\widehat{\text{supp}}(p(\mathbf{z}_c|Z_l^i))\backslash \widehat{\text{supp}}_c). \tag{3.20}$$

To prevent the problem we mentioned in the case where we completely abandon the regularization that the generative mechanism for one label may not be able to make sense of the latent variables for another label, we add the counterfactual image generation process to the model

training

$$\begin{aligned}\log p(\mathbf{z}_l^\dagger|\mathbf{x}_{\mathbf{z}_l^\dagger}, \mathbf{z}_l) &= \int p(\mathbf{x}_{\mathbf{Z}_l=\mathbf{z}_l^\dagger}|\mathbf{x}, \mathbf{z}_l) \log p(\mathbf{Z}_l = \mathbf{z}_l^\dagger|\mathbf{x}_{\mathbf{Z}_l=\mathbf{z}_l^\dagger}) d\mathbf{x}_{\mathbf{Z}_l=\mathbf{z}_l^\dagger} \\ &= \iint p(\mathbf{x}_{\mathbf{Z}_l=\mathbf{z}_l^\dagger}|\mathbf{z}_c) p(\mathbf{z}_c|\mathbf{x}, \mathbf{z}_l) \log p(\mathbf{Z}_l = \mathbf{z}_l^\dagger|\mathbf{x}_{\mathbf{Z}_l=\mathbf{z}_l^\dagger}) d\mathbf{z}_c d\mathbf{x}_{\mathbf{Z}_l=\mathbf{z}_l^\dagger},\end{aligned} \tag{3.21}$$

where $\mathbf{x}_{\mathbf{Z}_l=\mathbf{z}_l^\dagger}$ is the counterfactual image generated with a manually intervened label, $\mathbf{z}_l^\dagger \neq \widehat{\mathbf{z}}_l$ is the new label we change to, which can also be implemented as a stochastic policy, $\widehat{\mathbf{z}}_l$ being the most likely predicted label inferred from $\mathbf{x}$. The model is encouraged to correctly generate and classify the counterfactual number images. If this goal is well achieved, then the model is likely to know how to generate counterfactual images with combinations of $\mathbf{z}_c$ and $\mathbf{z}_l$ that are not commonly seen in the original dataset.

Finally, people may wonder, is it really okay if we don't give $\mathbf{z}_c$ a proper prior? Consider an example, shown in Figure 3.2. The data is sampled from a model where X and Y are dependent. In fact, let's assume $X = Y$, $Y \sim \mathcal{N}(0, 1)$. If we look at the sampled data, X and Y are going to be perfectly correlated, however, if we use a model, where X and Y are independent, to fit the data, the model is going to learn the marginal distribution for the two variables independently, i.e., $X \sim \mathcal{N}(0, 1)$, $Y \sim \mathcal{N}(0, 1)$ and $X \perp Y$. Even after the fitting has reached its optimum, the correlation of X and Y will not manifest in the model, because we did not give the model the capacity to record such correlations. However, this does not mean the conditional posterior is jointly independent, i.e., $p(x, y|\mathcal{D}) \neq p(x|\mathcal{D})p(y|\mathcal{D})$, where $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^N$ is the dataset, the information about dependence is still preserved in the dataset. So it is completely reasonable to model only a part of the probabilistic relationships and leave the part where dependence relationships are viciously hard to model to the dataset. This will be a huge issue for a generative model where we need to generate, e.g., x and y with realistic dependencies, if we don't have those dependencies in our model, then we can not generate tuple (x, y) that is close to the realistic cases. But this is not a problem for us since the generative model in our case is only utilized to generate counterfactual images where Z_c are inferred from real data, and we do not need to sample Z_c from a noise distribution.

After these modifications and the additional counterfactual task, we have our final VAE optimization goal:

$$\arg\max_{\theta,\phi,\psi} \int q(\mathbf{z}_c|\mathbf{x},\mathbf{z}_l)\log p(\mathbf{x}|\mathbf{z}_c,\mathbf{z}_l)d\mathbf{z}_c + \log p(\mathbf{z}_l|\mathbf{x}) + \sum_i m(\text{supp}(p(\mathbf{z}_c|Z_l^i))\backslash\text{supp}_c) + \log p(\mathbf{z}_l^\dagger|\mathbf{x}_{\mathbf{z}_l^\dagger},\mathbf{z}_l). \tag{3.22}$$

For experiments, we found that simply using a normal distribution with a high variance as the prior is normally good enough, which is also able to relax the restrictions imposed on the posterior of $\mathbf{z}_c$. As a result, the KL divergence between $q(\mathbf{z}_c|\mathbf{x},\mathbf{z}_l)$ and $p(\mathbf{z}_c|\mathbf{z}_l)$ can be written as

$$M = D_{KL}(q(\mathbf{z}_c|\mathbf{x},\mathbf{z}_l)|p(\mathbf{z}_c|\mathbf{z}_l)) = -\frac{1}{2}\sum_{j=1}^{J}(1+\log(\sigma_j^2)-\log(\sigma^2)-\frac{\mu_j^2}{\sigma^2}-\frac{\sigma_j^2}{\sigma^2}), \tag{3.23}$$

where σ^2 is the variance of the prior, μ_j and σ_j are the parameters of the posterior inferred by the recognition model.

PROOF. Assuming $q(\mathbf{z}_c|\mathbf{x},\mathbf{z}_l) = \mathcal{N}(\mu,\sigma^2)$, μ_i and σ_i being the i-th entry. $p(\mathbf{z}_c|\mathbf{z}_l) = (N)(0,\hat{\sigma}^2)$, where $\hat{\sigma}\in\mathbb{R}^J$, $\hat{\sigma}_i = \sigma$. Then

$$\begin{aligned}\int q(\mathbf{z}_c|\mathbf{x},\mathbf{z}_l)\log p(\mathbf{z}_c|\mathbf{z}_l)d\mathbf{z}_c &= \int \mathcal{N}(\mu,\sigma^2)\log(N)(0,\hat{\sigma}^2)d\mathbf{z}_c \\ &= -\frac{J}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{j=1}^{J}(\mu_j^2+\sigma_j^2).\end{aligned}$$

Combining with

$$\begin{aligned}\int q(\mathbf{z}_c|\mathbf{x},\mathbf{z}_l)\log q(\mathbf{z}_c|\mathbf{x},\mathbf{z}_l)d\mathbf{z}_c &= \int \mathcal{N}(\mu,\sigma^2)\log\mathcal{N}(\mu,\sigma^2)d\mathbf{z}_c \\ &= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(1+\log\sigma_j^2),\end{aligned}$$

We have

$$D_{KL}(q(\mathbf{z}_c|\mathbf{x},\mathbf{z}_l)|p(\mathbf{z}_c|\mathbf{z}_l)) = \int q(\mathbf{z}_c|\mathbf{x},\mathbf{z}_l)\log(q(\mathbf{z}_c|\mathbf{x},\mathbf{z}_l) - p(\mathbf{z}_c|\mathbf{z}_l))d\mathbf{z}_c$$
$$= -\frac{1}{2}\sum_{j=1}^{J}(1+\log\sigma_j^2 - \log\sigma^2 - \frac{\mu_j^2}{\sigma^2} - \frac{\sigma_j^2}{\sigma^2}).$$

□

3.3.4 Assessing PN and PS simultaneously

By combining these two notions, we also have *probability of necessity and sufficiency* (PNS), which is defined as

DEFINITION 3.3.3 (Probability of Necessity and Sufficiency, PNS). *The probability of necessity and sufficiency is defined as*

$$\begin{aligned}\text{PNS} =& P(B_{A=1}=1, B_{A=0}=0) && (3.24)\\ =& P(B_{A=1}=1|A=0,B=0)P(A=0,B=0)\\ &+ P(B_{A=0}=0|B=1,A=1)P(A=1,B=1) && (3.25)\\ =& P(A=0,B=0)\text{PS} + P(A=1,B=1)\text{PN}, && (3.26)\end{aligned}$$

which is a weighted average of PN and PS, depending on which situation happens more often. Note that events where $A=0, B=1$ or $A=1, B=0$ will not contribute to the term at all, because, in those situations, we can not assess how A and B are causally related.

To find a good causal representation, from the perspective of the discriminative task of classification, we need to find variables that have a high PNS on the resulting predicted label. However, since the discriminative model is only a proximation to the posterior probability, we reframe this as the probability of causation of the label to the variables. For instance, if we want a representation that can sufficiently determine the predicted label, we should look for variables to which, in the generative process, the root label has a high probability of necessity. That is, whenever the label is turned on, the variable is turned on. On the other hand, if we

want to find the necessary causes of the label in the discriminative task, we are essentially looking for variables to which the label is a sufficient cause. As a result, a representation that is a sufficient cause to the label is a variable R with a high probability of necessity for the label Z_l to R, i.e.,

$$\mathrm{PN} = P(R_{Z_l=0} = 0 | R = 1, Z_l = 1) \tag{3.27}$$

Usually, we are not able to directly assess the quantity of these counterfactual terms since we often do not have access to modifying the generative process. Thanks to the dual generative model trained in the disentangled VAE, we are able to estimate the counterfactual terms. Take PN as an example; by integrating over data $\mathbf{x}$, we have

$$\mathrm{PN} = \int P(R_{Z_l=0} = 0 | R = 1, Z_l = 1, \mathbf{x}) p(\mathbf{x} | R = 1, Z_l = 1) d\mathbf{x}. \tag{3.28}$$

3.3.5 Evaluation of Individual POC

Now we can focus on the individual term $P(R_{Z_l=0} = 0 | R = 1, Z_l = 1, \mathbf{x}^{(i)}) \coloneqq \mathrm{PN}_{\mathbf{x}}$. By integrating over the context latent space Z_c, we have

$$\begin{aligned}
\mathrm{PN}_{\mathbf{x}} &= \int P(R_{Z_l=0} = 0 | R = 1, Z_l = 1, \mathbf{x}, \mathbf{z}_c) P(\mathbf{z}_c | R = 1, Z_l = 1, \mathbf{x}) d\mathbf{z}_c \\
&= \int P(R = 0 | do(Z_l = 0), \mathbf{z}_c) P(\mathbf{z}_c | R = 1, Z_l = 1, \mathbf{x}) d\mathbf{z}_c && (3.29) \\
&= \iint P(R = 0 | \mathbf{x}_{Z_l=0}) P(\mathbf{x}_{Z_l=0} | Z_l = 0, \mathbf{z}_c) P(\mathbf{z}_c | R = 1, Z_l = 1, \mathbf{x}) d\mathbf{x}_{Z_l=0} d\mathbf{z}_c && (3.30) \\
&= \iint (1 - f(\mathbf{x}_{Z_l=0})) P(\mathbf{x}_{Z_l=0} | Z_l = 0, \mathbf{z}_c) P(\mathbf{z}_c | R = 1, Z_l = 1, \mathbf{x}) d\mathbf{x}_{Z_l=0} d\mathbf{z}_c. && (3.31)
\end{aligned}$$

Following the same procedure, we can also evaluate the counterfactual term PS conditioned on $\mathbf{x}$ as

$$\begin{aligned}
\mathrm{PS}_{\mathbf{x}} &= \int P(R_{Z_l=1} = 1 | R = 0, Z_l = 0, \mathbf{x}, \mathbf{z}_c) P(\mathbf{z}_c | R = 0, Z_l = 0, \mathbf{x}) d\mathbf{z}_c \\
&= \iint P(R = 1 | \mathbf{x}_{Z_l=1}) P(\mathbf{x}_{Z_l=1} | Z_l = 1, \mathbf{z}_c) P(\mathbf{z}_c | R = 0, Z_l = 0, \mathbf{x}) d\mathbf{x}_{Z_l=1} d\mathbf{z}_c && (3.32) \\
&= \iint f(\mathbf{x}_{Z_l=1}) P(\mathbf{x}_{Z_l=1} | Z_l = 1, \mathbf{z}_c) P(\mathbf{z}_c | R = 0, Z_l = 0, \mathbf{x}) d\mathbf{x}_{Z_l=1} d\mathbf{z}_c. && (3.33)
\end{aligned}$$

The PNS conditioned on individual data, on the other hand, is a little tricky. Because of the binary nature of events/variables in our model, we can expand $\mathrm{PNS}_{\mathbf{x}}$ as

$$\begin{aligned}
\mathrm{PNS}_{\mathbf{x}} =& P(R_{Z_l=0}=0, R_{Z_l=1}=1|\mathbf{x}) && (3.34)\\
=& P(R_{Z_l=0}=0, R_{Z_l=1}=1|R=1, Z_l=1, \mathbf{x})P(R=1, Z_l=1|\mathbf{x}) && (3.35)\\
&+ P(R_{Z_l=0}=0, R_{Z_l=1}=1|R=1, Z_l=0, \mathbf{x})P(R=1, Z_l=0|\mathbf{x}) && (3.36)\\
&+ P(R_{Z_l=0}=0, R_{Z_l=1}=1|R=0, Z_l=1, \mathbf{x})P(R=0, Z_l=1|\mathbf{x}) && (3.37)\\
&+ P(R_{Z_l=0}=0, R_{Z_l=1}=1|R=0, Z_l=0, \mathbf{x})P(R=0, Z_l=0|\mathbf{x}) && (3.38)\\
=& P(R_{Z_l=0}=0|R=1, Z_l=1, \mathbf{x})P(R=1, Z_l=1|\mathbf{x}) && (3.39)\\
&+ 0 && (3.40)\\
&+ 0 && (3.41)\\
&+ P(R_{Z_l=1}=1|R=0, Z_l=0, \mathbf{x})P(R=0, Z_l=0|\mathbf{x}) && (3.42)\\
=& \mathrm{PN}_{\mathbf{x}}P(R=1, Z_l=1|\mathbf{x}) + \mathrm{PS}_{\mathbf{x}}P(R=0, Z_l=0|\mathbf{x}) && (3.43)\\
=& \mathrm{PN}_{\mathbf{x}}f_R(\mathbf{x})\mathbb{I}_{Z_l=1} + \mathrm{PS}_{\mathbf{x}}(1-f_R(\mathbf{x}))\mathbb{I}_{Z_l=0}, && (3.44)
\end{aligned}$$

where $f_R(\mathbf{x})$ is the functional mapping that calculates the value of the representation R given $\mathbf{x}$ as the input. As we can see, depending on the label Z_l for an $\mathbf{x}$, the $\mathrm{PNS}_{\mathbf{x}}$ will collapse to a weighted $\mathrm{PN}_{\mathbf{x}}$ or $\mathrm{PS}_{\mathbf{x}}$.

3.3.6 Evaluation of Conditional POC

Quite often, we only want to assess the POC of an entry R_i in a multivariable representation $\mathbf{R}$. We also extend the definition of POC to conditional POC:

DEFINITION 3.3.4 (Conditional Probability of Necessity and Sufficiency). *The* PNS *of* Z_l *to* R_i *conditioned on* R_{-i} *is defined as*

$$\mathrm{PNS} = P(R_{i\,Z_l=1}=1, R_{i\,Z_l=0}=0|\mathbf{R}_{-i}=\mathbf{r}_{-i}). \tag{3.45}$$

This evaluates how one particular representation R_i responds to the intervention on Z_l while holding all other representations fixed.

3.3.7 From POC to Causal Representation Learning

Given the counterfactual goal, now we can use it to tame the learning of f_ψ. If there are L categories for $\mathbf{Z}_l$, we can set up the structure of f_ψ in parallel such that it has L leaf nodes $\mathbf{V}_o$ representing the predicted label. For example, if f_ψ is a NN parameterized by ψ, we can have a L-entry vector $\mathbf{V}_o$ as the output layer, while $\mathbf{V}$ represents all variables/nodes involved in the NN. Since we want the predicted label to be causally faithful to the true label, the overall counterfactual goal is

$$\text{PNS} = \prod_j^L P(\mathbf{V}_{o,j\,Z_{l,j}=0} = 0, \mathbf{V}_{o,j\,Z_{l,j}=1} = 1 | \mathbf{V}_{o,-j} = \mathbf{v}_{o,-j}), \tag{3.46}$$

where $\mathbf{Z}_l$ is a one-hot vector representing the label with $\sum_i \mathbf{Z}_{l,i} = 1$. $\mathbf{Z}_{l,i} \in \{0, 1\}$ represents where the data belongs to label i.

If f_ψ accidentally picks up a spurious feature, like color in the colored MNIST case, it will not contribute to the final predicted label $\mathbf{V}_o$ since it will be proved spurious with counterfactual assessment. But we also need an incentive for f_ψ to drop these spurious features, because they can be included in $\mathbf{V}$ without being actually used for inferring labels. Thus we suggest adding a normalization term $||\psi||$ to phase out these spurious features and only preserve those that can contribute causally stable/non-spurious information.

3.4 Optimization

Combining the previous losses, the overall optimization goal is

$$\mathcal{L} + \log p(\mathbf{z}_l | \mathbf{x}) + \log p(\mathbf{z}_l^\dagger | \mathbf{x}_{\mathbf{z}_l^\dagger}, \mathbf{z}_l) + \text{PNS} + ||\psi||. \tag{3.47}$$

As a common way to implement the VAE, we use NNs and Gaussians to describe the generative model $p_\theta(\mathbf{x}|\mathbf{z})$ and the recognition model $q_\phi(\mathbf{z}|\mathbf{x})$ as

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(f_\theta(\mathbf{z}), \mathbf{I})(\mathbf{x}) \tag{3.48}$$

$$q_\phi(\mathbf{z}|\mathbf{x}) = q_{\phi,c}(\mathbf{z}_c|\mathbf{x})q_{\phi,l}(\mathbf{z}_l|\mathbf{x}) \tag{3.49}$$

$$= \mathcal{N}(\mu_\phi(\mathbf{x}), \sigma^2_\phi(\mathbf{x}))(\mathbf{z}_c)\text{Cat}(f_{\phi,l}(\mathbf{x}))(\mathbf{z}_l), \tag{3.50}$$

where $\mathcal{N}(\mu, \sigma)(\mathbf{x})$ is a multivariate Gaussian, $f_\theta, \mu_\phi, \sigma^2_\phi, f_{\phi,l}$ are all NNs. $\mathbf{z}_c, f_{\phi,l(\mathbf{x})_i}(\mathbf{x}) \in \mathbb{R}^L$, $f_{\phi,l(\mathbf{x})_i}$ is normalized into a probability distribution by softmax. Cat is a categorical distribution such that $\sum_i \text{Cat}(i) = 1$ and $p(i) = f_{\phi,l(\mathbf{x})_i}$.

One additional thing we need to address is the exogenous variables. If you look at a standard structural causal model, every node is equipped with an exogenous noise to explain its individual unique randomness, but in our architecture, only the end layer $\mathbf{Z}_c$, $\mathbf{Z}_l$, and $\mathbf{X}$ are equipped with randomness, all intermediate nodes within the NN are deterministic given their parents. You may ask if there is a model that can be described by an SCM but not our framework. We argue that our framework is equivalently general in the sense that if there is a structural causal model associated with graph G, then we can construct a causal model with G' using our framework where only the root and leaf nodes have randomness. This is obvious because if you have an exogenous variable you want to use in the middle of the NN, you can relocate it to the root layer and pass it through the previous layers unchanged and deliver it to the place where you need it.

3.4.1 Optimizing Disentangled VAE

3.4.1.1 Gumbel-Softmax Approximation to Categorical Distribution

Given the recognition model $q_\phi(\mathbf{z}|\mathbf{x})$, by marginalize over $\mathbf{z}_c$, we have the marginal distribution of $p(\mathbf{z}_l|\mathbf{x}) = \int q_\phi(\mathbf{z}_c, \mathbf{z}_l|\mathbf{x})d\mathbf{z}_c$. Because we use a NN to approximate the value of $\mathbf{z}_l$, $q_\phi(\mathbf{z}|\mathbf{x})$ can be expanded as $q_\phi(\mathbf{z}_l|\mathbf{x}) = \text{Cat}(\text{Softmax}(\mathbf{z}^*_l(\mathbf{x};\phi)))$, where $\mathbf{z}^*_l(\mathbf{x};\phi)$ is the last output layer of the NN. An issue with this implementation is that the direct output prediction $\mathbf{z}^*_l$ will not be exactly a one-hot vector, which may produce unrealistic samples with these

non-one-hot $\mathbf{z}_l$. Of course, we can get a one-hot identity vector by using $\arg\max$, but we want the counterfactual goal we proposed in Equation 3.21 to propagate back gradients to $\widehat{\mathbf{z}_l}$, and thereafter q_ϕ. Thus we decide to use the Gumbel-Softmax [18] distribution to synthesize a close-to-one-hot vector while still preserving the differentiability.

DEFINITION 3.4.1 (Gumbel-Softmax Reparameterization). *Let $\mathbf{z}_l^*$ be the direct output of the recognition model $q_\phi(\mathbf{z}_l|\mathbf{x})$. The approximation to the one-hot $\mathbf{z}_l$ is defined as $\widehat{\mathbf{z}_l}$ such that*

$$\widehat{\mathbf{z}_{l,i}} = \frac{\exp((\log \mathbf{z}_{l,i}^* + g_i)/\tau)}{\sum_i \exp((\log \mathbf{z}_{l,i}^* + g_i)/\tau)}, \tag{3.51}$$

where g_i are sampled from the standard Gumbel$(0, 1)$ distribution.

$\widehat{\mathbf{z}_{l,i}}$ is very close to one-hot (depending on the hyperparameter τ) while still preserves differentiable path to $\mathbf{z}_l^*$.

3.4.1.2 Monte Carlo Estimation of ELBO

By adopting the reparameterization, we can expand $q(\mathbf{z}_c|\mathbf{x}, \mathbf{z}_l)$ as $\mathcal{N}(\mu_\phi(\mathbf{x}), \sigma_\phi^2(\mathbf{x}))$, where μ_ϕ and σ_ϕ are implemented as a NN. Via Monte Carlo integration, the ELBO can be estimated by

$$\hat{\mathcal{L}} = \sum_i^N (\mu_\phi(\mathbf{x}) + \sigma_\phi(\mathbf{x})\epsilon_i) \log p(\mathbf{x}|\mathbf{z}_c, \mathbf{z}_l) - D_{KL}(q(\mathbf{z}_c|\mathbf{x}, \mathbf{z}_l)|p(\mathbf{z}_c|\mathbf{z}_l)), \tag{3.52}$$

where $\epsilon_i \sim \mathcal{N}(0, 1)$, N is the number of samples that we use for Monte Carlo integration. The second term of KL divergence usually has an analytical form.

3.4.2 Evaluating Counterfactual terms

3.4.2.1 Evaluation Over a Dataset

The integration over $\mathbf{x}$ can be estimated by evaluating all samples in the dataset. For example, the conditional counterfactual term PN can be estimated by

$$\widehat{\mathrm{PN}} = \frac{1}{N}\sum_{i}^{N} P(R_{Z_l=0}=0|R=1,Z_l=1,\mathbf{x}^{(i)})\mathbb{I}_{R=1,Z_l=1}. \tag{3.53}$$

$\mathbb{I}_{R=1,Z_l=1}$ is the indicator function that $\mathbb{I}(\mathbf{x}^{(i)},\mathbf{z}_l^{(i)})=1$ whenever $R=f(\mathbf{x})=1$ and $z_l=1$ and $\mathbb{I}=0$ o.w..

3.4.2.2 Assessment of Conditional Probability of Causation

To estimate the conditional probability of causation term, like $\mathrm{PN}_{\mathbf{x}}$, can also be estimated by sampling $\widehat{\mathbf{z}_c}$ over $q_\phi(\mathbf{z}_c|\mathbf{x},\mathbf{z}_l)$ and $\hat{\mathbf{x}}$ over $p_\theta(\mathbf{x}|\widehat{\mathbf{z}_c})$. As a result, we have the Monte Carlo estimated $\mathrm{PN}_{\mathbf{x}}$ as

$$\widehat{\mathrm{PN}_{\mathbf{x}}} = \frac{1}{N}\sum_{i=1}^{N}(1-f_R(\hat{\mathbf{x}}_{Z_l=0,i})), \tag{3.54}$$

where f_R is the functional mapping from $\mathbf{x}$ to the representation of interest R, $\hat{\mathbf{x}}_{Z_l=0,i}\sim p_\theta(\mathbf{x}_{Z_l=0}|Z_l=0,\widehat{\mathbf{z}_{c,i}})$, $\widehat{\mathbf{z}_{c,i}}\sim q_\phi(\mathbf{z}_c|R=1,Z_l=1,\mathbf{x})$, N is also the number of samples that we use for the estimation. Using the same technique, we can get the estimation for $\widehat{\mathrm{PS}_{\mathbf{x}}}$ as

$$\widehat{\mathrm{PS}_{\mathbf{x}}} = \frac{1}{N}\sum_{i=1}^{N} f_R(\hat{\mathbf{x}}_{Z_l=1,i}), \tag{3.55}$$

as well as $\widehat{\mathrm{PNS}_{\mathbf{x}}}$:

$$\widehat{\mathrm{PNS}_{\mathbf{x}}} = \frac{1}{N}\sum_{i=1}^{N}\left[(1-f_R(\hat{\mathbf{x}}_{Z_l=0,i}))f_R(\mathbf{x})\mathbb{I}_{Z_l=1}+f_R(\hat{\mathbf{x}}_{Z_l=1,i})(1-f_R(\mathbf{x}))\mathbb{I}_{Z_l=0}\right], \tag{3.56}$$

3.4.2.3 Counterfactual Recognition Goal

Similarly, the counterfactual goal $\log p(\mathbf{z}_l^\dagger|\mathbf{x}_{\mathbf{z}_l^\dagger},\mathbf{z}_l)$ in the VAE framework to enhance the unified semantic space can also be estimated via sampling the counterfactual images $\mathbf{x}_{\mathbf{z}_l^\dagger}$ from

its posterior. Eventually, we have

$$\widehat{\log p(\mathbf{z}_l^\dagger|\mathbf{x}_{\mathbf{z}_l^\dagger}, \mathbf{z}_l)} = \sum_i^N \log q_\phi(\mathbf{Z}_l = \mathbf{z}_l^\dagger|\hat{\mathbf{x}}_{\mathbf{Z}_l=\mathbf{z}_l^\dagger,i}), \tag{3.57}$$

where $\hat{\mathbf{x}}_{\mathbf{Z}_l=\mathbf{z}_l^\dagger,i} \sim p_\theta(\mathbf{x}|\mathbf{Z}_l = \mathbf{z}_l^\dagger, \widehat{\mathbf{z}_{c,i}})$, $\widehat{\mathbf{z}_{c,i}} \sim q_\phi(\mathbf{z}_c|R = 1, Z_l = 1, \mathbf{x})$.

3.4.3 Overall Optimization Procedure

When we apply stochastic gradient descent to these Monte Carlo estimations, a lot of the sampling procedure can be integrated into the sampling of the dataset. Eventually, we are able to compose a causal representation learning algorithm as Algorithm 1.

Algorithm 1 Causal Representation Learning

Require: $\{\mathbf{x}^{(i)}, \mathbf{z}_l^{(i)}\}_i^N$, $f : x \to v$
  initialize p_θ, q_ϕ
  while not converged **do**
    Loss $= 0$
    sample $\mathbf{x}$ and $\mathbf{z}_l$ from the dataset
    feed $\mathbf{x}$ into the recognition model to get $\mu_\phi(\mathbf{x})$, $\sigma_\phi^2(\mathbf{x})$, and $\mathbf{z}_l^*(\mathbf{x})$
    sample ϵ from $\mathcal{N}(0, 1)$
    $\widehat{\mathbf{z}_c} = \mu_\phi(\mathbf{x}) + \sigma_\phi(\mathbf{x})\epsilon$
    $\log p(\mathbf{z}_l|\mathbf{x}) = \text{CrossEntropy}(\mathbf{z}_l^*, \mathbf{z}_l)$
    $\widehat{\mathbf{z}_l} = \text{Gumbel-Softmax}(\mathbf{z}_l^*(\mathbf{x}))$
    $\mathbf{x}^* \sim p_\theta(\mathbf{x}|\widehat{\mathbf{z}_l}, \widehat{\mathbf{z}_c})$
    $\mathcal{L} = \text{MSE}(\mathbf{x}^*, \mathbf{x}) + M$
    sample a counterfactual label $\mathbf{z}_l^\dagger$ from a stochastic policy
    $\hat{\mathbf{x}}_{\mathbf{Z}_l=\mathbf{z}_l^\dagger} \sim p_\theta(\mathbf{x}|\mathbf{z}_l^\dagger, \widehat{\mathbf{z}_c})$
    feed the counterfactual image $\hat{\mathbf{x}}_{\mathbf{Z}_l=\mathbf{z}_l^\dagger}$ to the recognition model to get $\mathbf{z}_l^{\dagger *}(\hat{\mathbf{x}}_{\mathbf{Z}_l=\mathbf{z}_l^\dagger})$
    $\log p(\mathbf{z}_l^{\dagger *}|\mathbf{x}_{\mathbf{z}_l^\dagger}) = \text{CrossEntropy}(\mathbf{z}_l^{\dagger *}, \mathbf{z}_l^\dagger)$
    $\widehat{\text{PNS}_\mathbf{x}} = (1 - f_R(\hat{\mathbf{x}}_{Z_l=0}))f_R(\mathbf{x})\mathbb{I}_{Z_l=1} + f_R(\hat{\mathbf{x}}_{Z_l=1})(1 - f_R(\mathbf{x}))\mathbb{I}_{Z_l=0}$
    Loss $= \log p(\mathbf{z}_l|\mathbf{x}) + \mathcal{L} + \log p(\mathbf{z}_l^{\dagger *}|\mathbf{x}_{\mathbf{z}_l^\dagger}) + \widehat{\text{PNS}_\mathbf{x}} + ||\psi||$
    $\phi, \theta, \psi \leftarrow \nabla_{\phi,\theta,\psi}$Loss
  end while

Every time a piece of data is fed into the network, the ELBO is calculated, as well as the relaxed regularization on $p(\mathbf{z}_c|\mathbf{x}, \mathbf{z}_l)$. For the counterfactual goal, we flipped the label to another one and use the $\mathbf{z}_c$ inferred from this data to generate counterfactual data and feed

it to the recognition model. Then, we calculate the probability of causation as well as the norm of f_ψ and add all these goals together and differentiate it with regards to the parameters of the generative model p_θ, the recognition model p_ϕ and the feature extraction function f_ψ respectively.

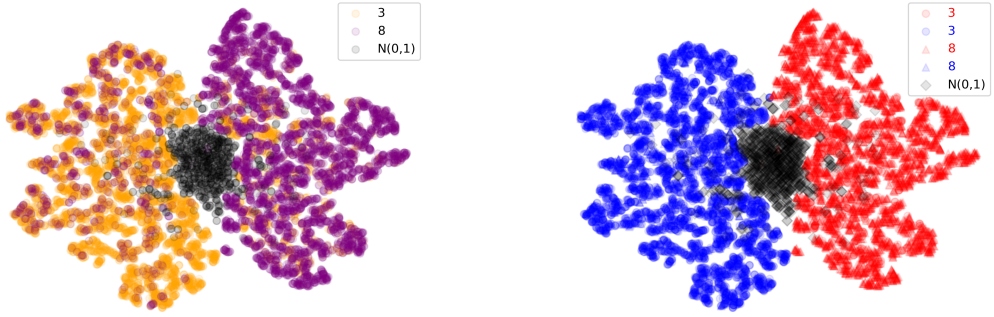
3.5 Experiments

In this section, we use various datasets and tasks to demonstrate that our framework is capable of producing counterfactual images with consistent semantic characteristics. As a result, representations learned from our framework are able to perform well on o.o.d. samples. In Section 3.5.1, we visualize the counterfactual images generated on dataset colored MNIST and CelebA and demonstrate that our model has learned the unified semantic space. In Section 3.5.2, we show the quantitative results tested on color edMNIST and CelebA to verify that our representation tolerates o.o.d. samples.

All experiments are conducted on a machine with 4×RTX3080 graphic cards, all datasets are publicly available.

3.5.1 Counterfactual Image Generation

As we have shown in Figure 3.3a, the tSNE embedding of $\mathbf{z}_c$ indicates that because of the inconsistent semantic space, the intervention of the label may also change the color of the number along with the identity of the number. After adopting our VAE framework with relaxed posterior regularization and counterfactual recognition goal, we found that our model can effectively disentangle the identity and other characteristics in terms of semantics. Note that this does not mean $\mathbf{Z}_c \perp \mathbf{Z}_l$, it means that no matter which labels $\mathbf{z}_l$ is given, $\mathbf{Z}_c$ represent the same semantic information. In fact, the independence of $\mathbf{Z}_c$ and $\mathbf{Z}_l$ and a unified semantic space are two mutually exclusive things. Data will have different semantic distributions when given different labels, for example, given the person is male, the distribution of hairstyle is certainly very different from when given the person is female, albeit they have the same



(A) $p(\mathbf{Z}_c|Z_l)$ (B) $p(\mathbf{Z}_c|Z_l, \text{color})$

FIGURE 3.6. **tSNE embedding for the latent space in our model.** A VAE is trained on the colored MNIST dataset with relaxed prior regularization and the counterfactual goal $\log p(\mathbf{z}_l^\dagger|\mathbf{x}_{\mathbf{z}_l^\dagger}, \mathbf{z}_l)$, we illustrate the distribution of $\mathbf{Z}_l$ conditioned on attributes. 3.6a shows that with relaxed prior regularization, the conditional distributions $p(\mathbf{Z}_l|Z_l)$ although have the same support (for any part of $\mathbf{Z}_c$, there are $\mathbf{z}_c$ inferred from both number 3 and the number 8), are quite different in terms of probability density, which is expected if the space of $\mathbf{Z}_c$ has the same unified semantic space, just like the conditional distributions of color given different labels are different. 3.6b further affirms that $\mathbf{Z}_c$ has been embedded in a unified semantic space, where $\mathbf{z}_c$s close to each other also have similar attributes, like color, which is why the distribution/space of $\mathbf{Z}_c$ can be well separated by color. This implies that our model is capable of producing counterfactual images by intervening on the label that will preserve identity-unrelated attributes like color.

support, meaning that the ability to have a certain type of hairstyle is irrelevant to the gender but certainly male are more inclined to certain styles and female are inclined to certain others.

We replot the tSNE embedding of $\mathbf{Z}_c$ for our model, shown in Figure 3.6. From Figure 3.6a, we can see that after the posterior regularization is relaxed, the support of $p(\mathbf{z}_c|\mathbf{z}_l)$ is the same no matter which labels $\mathbf{z}_l$ is given (noticing that their regions are overlapped with each other), but the density difference is whopping. We can see that 3s are more likely to appear on the left side and 8s are more likely to appear on the right side, which is expected if they were to have a unified semantic space because they have different color distributions. More importantly, as shown in Figure 3.6b, numbers of different colors do not overlap with each other anymore, which means that a $\mathbf{Z}_c$ inferred from, for example, a blue 3 is also going to

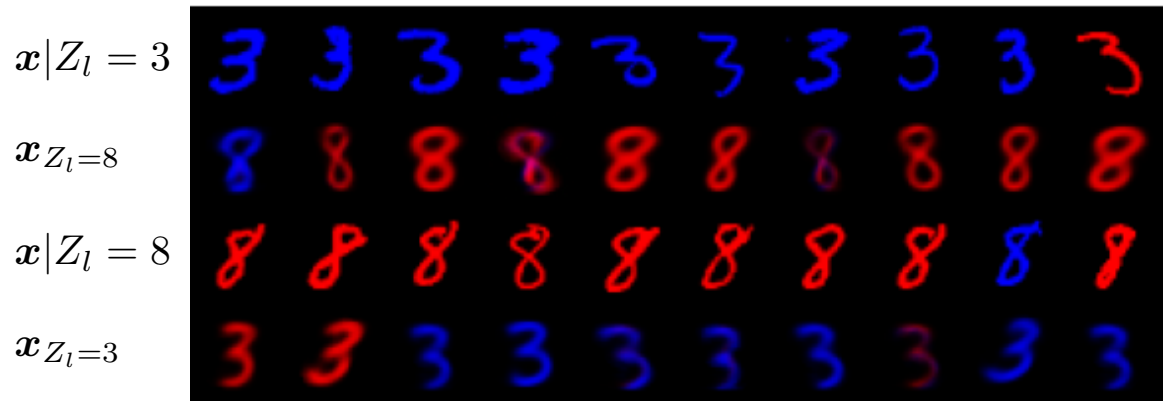


FIGURE 3.7. **Counterfactual results of CVAE on colored MNIST.** Because the regular CVAE uses the standard Gaussian as the prior for the latent space, it is not able to disentangle the semantic information from the identity, the same $\mathbf{z}_c$ that causes the image to be blue for number 3, for example, may also cause the image to be red when the number is 8. As a result, the counterfactual images generated by changing labels are also going to change the colors.

work fine to generate a blue 8 when Z_l is intervened to take the value of 8. This will only happen when the regions of $\mathbf{Z}_c$ of blue numbers and red numbers are separated, which proves that the latent space of our model has universal semantic meanings.

To illustrate this more directly, we show the counterfactual images generated by our model on different datasets.

3.5.1.1 Colored MNIST

We study the counterfactual image on colored MNIST [1]. The colored MNIST dataset is constructed from the original MNIST dataset by coloring the numbers with a stochastic policy so that numbers are correlated with certain colors. We focus on the numbers 3 and 8 with the colors red and blue.

To create a training set, we color the images of 3 blue with a possibility of p and red with a possibility of $1 - p$. For the images of number 8, the possibility is reversed. We color images blue with a possibility of $1 - p$ and red with a possibility of p, $p \in (0, 1)$. When p is close to 1, images of 3 are mostly colored blue, so that 3 has a high correlation with the color blue. The relationship between correlation and p is $\rho(Z_l = 3, \text{blue}) = 2p - 1$, where ρ is the

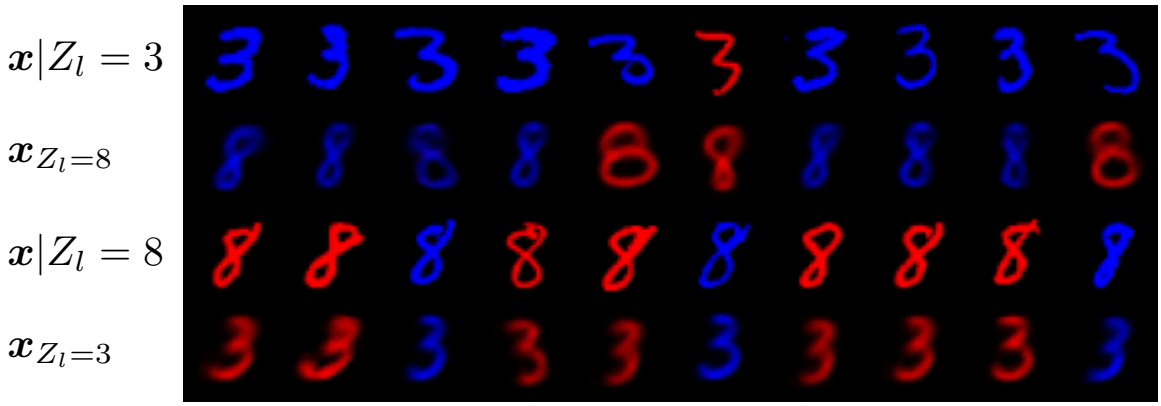


FIGURE 3.8. **Counterfactual results of our model on colored MNIST.** Our framework relaxes the KL divergence regularization imposed on the posterior $p(\mathbf{z}_c|\mathbf{z}_l)$ and use a counterfactual recognition goal to encourage the model to learn a unified latent space with universal semantic meaning. This means that if such a semantic space is learned, and given a $\mathbf{z}_c$ that causes a number to be blue, no matter how you manipulate the label, the color of the counterfactual image is going to remain the same. As shown in the figure, our framework can consistently generate blue 8s with blue 3s or red 3s with red 8s, which are quite rare in the original training set. This is also the reason we call them counterfactual images because they are images that do not factually exist or rarely exist, facts refer to the i.i.d. distribution that we sampled our training dataset from.

correlation. For example, when $p = 0.9$, the correlation between the 3 and blue is 0.8; when $p = 0.5$, images are randomly colored with a 50-50 chance, so the correlation is 0. For this counterfactual image generation experiment, we set $p = 0.9$. To further increase the difficulty of classification, we randomly change the label of the training data with a 0.25 possibility. As a result, 25% of the training data have wrong labels.

By constructing such a training set, models that can not recognize the spurious correlation would consider the color as part of the features of that number, and the generation process of color would be conditioned on the label. As a result, for a vanilla conditional VAE that does not have any counterfactual guidance, the same Z_c that generates blue 3 would generate red 8 when the label is intervened, as shown in Figure 3.7, second row. Similarly, a red 8 would likely be transformed into a blue 3. When we talk about causality, essentially we are focusing on the ability to alter the current reality, which, in this case, is the data distribution $p(\mathbf{x})$. When p is high and the correlation is strong, most images of 3 are blue, we wish our

model can consistently generate red 3s from red 8s so that during representation learning it can understand color is not a feature of the number itself. This will not happen if $p(\mathbf{z}_c|\mathbf{z}_l)$ are the same across different $\mathbf{z}_l$, because the counterfactual data would conform to the same distribution, i.e., $p(\mathbf{x}_{Z_l=8}|Z_l = 3) = p(\mathbf{x}|Z_l = 8)$ and $p(\mathbf{x}_{Z_l=3}|Z_l = 8) = p(\mathbf{x}|Z_l = 3)$. As shown in Figure 3.7, the counterfactual 8s are mostly red, and counterfactual 3s are mostly blue as the training set. Such counterfactual images would be meaningless because we are only reaffirming the original data distribution instead of creating a new distribution with lots of red 3s and blue 8s that are rarely distributed in the original $p(\mathbf{x})$.

After relaxing the regularization on the posterior $p(\mathbf{z}_c|z_l)$ and using the counterfactual goal of Equation 3.21 to guide the model to learn a unified semantic space, our model has successfully disentangled the context information from the identity information. This does not mean our $\mathbf{Z}_c$ is independent of Z_l, on the contrary, they are dependent as we intended, but the semantic meaning of $\mathbf{Z}_c$ are now independent of Z_l. As a result, the same $\mathbf{z}_c$ that produces blue 3 is going to produce blue 8 in our model when Z_l is intervened to $Z_l = 8$, as shown in Figure 3.8, which is a rare kind in the original distribution. The counterfactual images synthesized from the real data has a completely different data distribution compared to the original one where images of 8 are mostly blue and images of 3 are mostly red.

It is also worthwhile to point out that we did not impose any supervision during the training other than the identity labels, which means that other aspects besides color are also disentangled from the identity. If we outline the tilt angle of the numbers, shown in Figure 3.9, we can see that the majority of them are preserved after the intervention, while for the vanilla CVAE in Figure 3.7, the tilt angles of counterfactual images are obviously unrelated to the original data.

3.5.1.2 CelebA

The coloring mechanism in colored MNIST is quite easy to understand and disentangled from the generation of (the shape of) the number, presumably, the process is also easy for models to learn. Now we turn to CelebA [24], which is a much harder and more realistic dataset with a higher resolution (128×128 vs. 28×28 for colored MNIST). We focus on

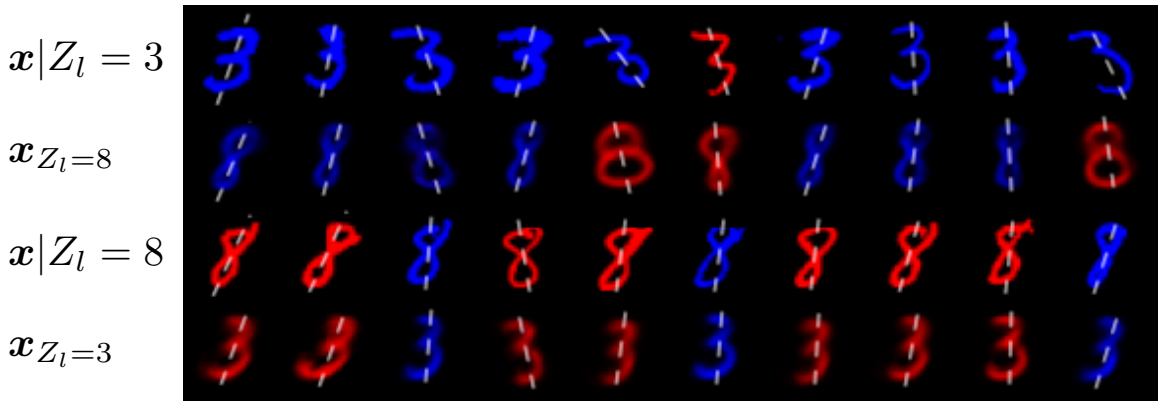


FIGURE 3.9. **Our counterfactual images with angles outlined.** It is worthwhile to point out that the label Z_l is the only variable that we supervise, meaning that attributes like color, distortion, etc. are automatically embedded into a unified semantic space. In fact, if you compare our counterfactual images with the ones with vanilla VAE in Figure 3.7, we can see that our model does a better job of preserving other semantic information like angles or distortion without supervision.

FIGURE 3.10. **Counterfactual results of CVAE on CelebA.** A sub-dataset of CelebA is constructed such that images of **men** are highly correlated with **smiling**, while images of **women** are highly correlated with **not smiling**, as shown above, labeled by $\mathbf{x}|Z_l = \text{female}$ and $\mathbf{x}|Z_l = \text{male}$. As we can see, a vanilla VAE picks up the spurious correlation and considers smiling as a feature of men and not-smiling as a feature of women. When we intervene on the label Z_l, the smiling will also be changed as part of the features of gender. As a result, non-smiling women will be transformed into smiling men, and vice versa, which will not help with our causal reasoning because the model is only reaffirming the original data distribution by generating more similar data with the same spurious correlation.

FIGURE 3.11. **Counterfactual results of our model on CelebA.** Our model is also tested on the same CelebA sub-dataset with the same spurious correlation. As we can see, our model is capable of disentangling the semantic information from the label, such that as long as we only change label Z_l, semantic information encoded in Z_c will not change. As a result, during the counterfactual image generation, an image of a non-smiling woman will be transformed into a man while preserving the non-smiling feature. Consequently, we can use our model to generate smiling women and non-smiling men consistently by intervening on the label Z_l of real data, which are rarely distributed in the original data distribution. Thus our model is able to generate meaningful counterfactual data that can help our causal reasoning.

two attributes, gender and smiling, for the reason that these two are the most obvious and unambiguous attributes in CelebA. Quite like how we construct a colored MNIST dataset, we create our correlated training set by sub-sampling from the full CelebA dataset. First, we divide all data into four categories, smiling men, non-smiling men, smiling women, and non-smiling women. We conduct sub-sampling for the 4 subsets so that the size satisfies the proportion $\text{size}(\text{smiling men}) : \text{size}(\text{non-smiling men}) : \text{size}(\text{smiling women}) : \text{size}(\text{non-smiling women}) = p : 1-p : 1-p : p$. The higher the p is, the higher the correlation between man and smiling and the correlation between woman and non-smiling are. At the same time, we also make sure data of different genders are balanced. For our task, we set $p = 0.9$. As a result, 90% of all men in the dataset are smiling, and 90% of women are not smiling. The vanilla CVAE easily pick up the correlation between gender and smiling, and assume that smiling is part of the features of men and non-smiling is a feature of women. When we intervene on the label Z_l, the resulting counterfactual image is going to have different smiles depending on the new gender. As shown in Figure 3.10, the vanilla CVAE fails to recognize that smiling is a spurious feature, the counterfactual images have

different smiling compared to the source images, i.e., non-smiling women are transformed into smiling men and smiling men are transformed into non-smiling women. As a result, the correlation between gender and smiling is preserved after the intervention on the label.

On the other hand, our model realizes that smiling is only a spurious feature, and gender is not defined by smiling. As shown in Figure 3.11, after the intervention, the smiling/non-smiling is preserved after the gender label is intervened. Thus, non-smiling women are transformed into non-smiling men and smiling men are transformed into smiling women. These counterfactual images are rarely distributed in the original dataset, and thus we are effectively generating images from a completely different distribution.

3.5.2 Representation Performance on o.o.d. Samples

In this section, we assess the performance of our causal representation via classification accuracy on out-of-distribution samples.

3.5.2.1 Colored MNIST

To quantitatively assess our framework with o.o.d. samples, we constructed a testing set symmetrical to the training set by coloring the images in an opposite way so that they have the opposite correlation compared to the training set. That is, we color images of 3 blue with a possibility $1-p$ and red with p, and color images of 8 blue with p and red with $1-p$. When p is close is 1, images of 3 are colored mostly red, which is the opposite of the training set. We apply the 0.25 chance random change on the testing labels as well, as a result, the theoretical upper bound for performance is 0.75. If a model fails to detect the spurious correlation in the training data, it will fail to perform well on the testing set. For instance, assume the model considers blue a good representation for images of 3 after looking at the high correlation of 3 and blue in the training set. When the model encounters a blue 8, it would think the number is 3, hence the classification accuracy would be bad.

We compare the classification performance of our model with CAUSAL-REP [57], vanilla VAE, and direct NN fit under various correlations of numbers and colors. As shown in Figure

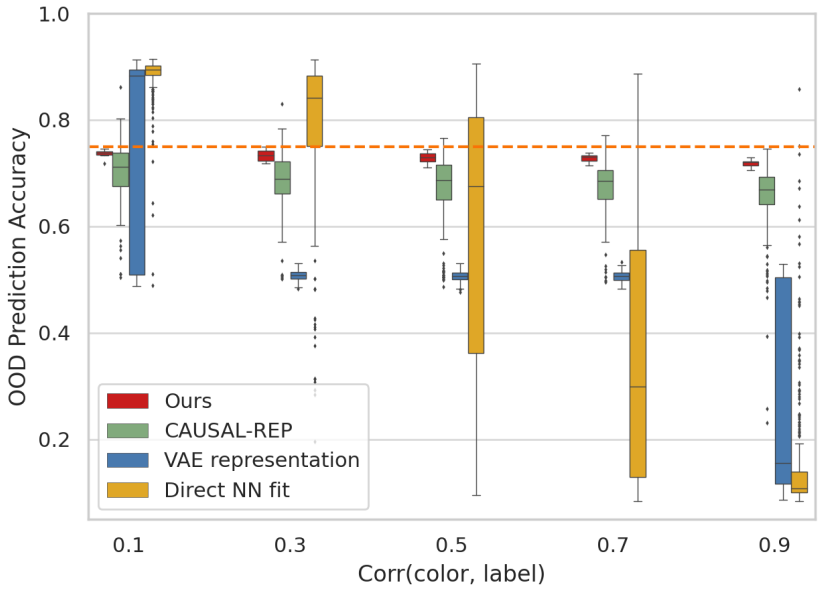


FIGURE 3.12. o.o.d. prediction results. By sampling from the original MNIST dataset, we created a training dataset of colored MNIST where the label is correlated with the color. For example, if Corr$(Z_l = 3, \text{color} = \text{blue}) = 0.9$, it means 95% of images of number 3 are blue, and also 95% of images of number 8 are red. For the test dataset, the correlation is oppositely constructed. In the previous example, the test dataset will have a high correlation between the number 3 and the color red and a high correlation between the number 8 and the color blue. The yellow bar (0.75) is the theoretical upper bound for performance because labels in the test set have a 25% chance of being messed up.

TABLE 3.1. o.o.d. prediction on Colored MNIST without flip on the testset

Corr(color, label)	0.1	0.3	0.5	0.7	0.9
Accuracy(%)	96.46 ± 0.36	96.47 ± 0.28	95.46 ± 0.65	93.99 ± 1.00	86.05 ± 2.26

3.12, we can see that our model outperforms other methods by a large margin, which indicates that representations learned from our framework do not include as many spurious features as other methods do. Table 3.1 also shows the classification accuracy of the representation when the 0.25 chance random flip is not applied.

TABLE 3.2. o.o.d. prediction on CelebA.

target label	spurious feature	Corr (train)	Corr (test)	Direct NN fit	vanilla VAE	CAUSAL-REP	Ours
Arched Eyebrows	Bags Under Eyes	0.80	−0.80	0.514	0.499	0.539	**0.6865**
Arched Eyebrows	Wearing Earrings	0.80	−0.80	0.504	0.494	0.521	**0.6135**
Attractive	Wearing Necklace	0.80	−0.80	0.505	0.485	0.537	**0.6682**
Black Hair	Mouth Slightly Open	0.80	−0.80	0.566	0.505	0.594	**0.7027**
Mustache	Black Hair	0.80	−0.80	0.512	0.540	0.525	**0.8027**

3.5.2.2 CelebA

Very much like the colored MNIST representation classification experiment. First, we select various pairs of attributes, for each pair, we set up a training set where the two attributes are positively correlated, while in the testing set, the two attributes are negatively correlated. The results are shown in Table 3.2, as we can see, our representation learned from our model is able to resist the spurious correlation and outperform other methods on the o.o.d. samples.

3.5.3 Generated Images via Sampling $\mathbf{Z}_c$ from Gaussian

As we have discussed, since we have relaxed the regularization on the conditional posteriors $p(\mathbf{z}_c|\mathbf{z}_l)$, it is impossible for them to have a unified semantic space and remain close to the standard Gaussian $\mathcal{N}(0,1)$ at the same time, and more likely than not, $\mathcal{N}(0,1)$ is going to be located near the border between the different posteriors. As a result, in our model, $\mathbf{z}_c$ sampled from $\mathcal{N}(0,1)$ will not resemble $\mathbf{z}_c$ conditioned on Z_l, and thus the images generated from our VAE are not going to look as realistic as the vanilla VAE, as shown in Figure 3.13. However, as we have pointed out, the purpose of our model is not to generate synthetic images, but to perform counterfactual image generation, taking $\mathbf{z}_c$ inferred from real data. Thus, the undermined generative ability is a cost that we have to pay but does not affect the purpose of our task, which is representation learning.

(A) Genrated samples from Vanilla VAE via $Z_c \sim \mathcal{N}(0,1)$

(B) Genrated samples from ours via $Z_c \sim \mathcal{N}(0,1)$

FIGURE 3.13. **Generated images from vanilla VAE and ours.** By relaxing the posterior distribution $p(\mathbf{z}_c|\mathbf{z}_l)$, we are able to learn a unified semantic space for $\mathbf{z}_c$, but one side effect is that the random noise sampled from $\mathcal{N}(0,1)$ are not going to be as close to the real $\mathbf{z}_c$s inferred from the real data as a vanilla VAE. However, considering that the whole generative model is only here to assist the counterfactual generation and causal reasoning, this side effect does not undermine our purpose.

3.6 Conclusion and Discussion

In this chapter, we proposed a causal representation learning framework along with a novel VAE capable of conducting counterfactual reasoning. For causal reasoning, we already have a set of concepts, the probability of causation, to evaluate how one binary event can causally affect another one using counterfactual reasoning. Of course, to do counterfactual reasoning, we have to model the generative process with the representation learning process, so we resort to VAE since it is a classic probabilistic generative model. We discovered that the issue with the traditional VAE is that it also models the spurious correlation because of the strict regularization imposed on the posterior of the latent variable, resulting in a latent space with inconsistent semantic meanings under different conditions. As a result, VAE fails to conduct correct counterfactual generation where we want the model to only change the identity to the label that we intervened on and leave the rest of the features unchanged. To fix this, we propose to relax the regularization on the posterior and propose to add a counterfactual task to encourage the formation of a unified semantic latent space. Once the model is capable of conducting counterfactual reasoning, we use it to evaluate the probability of causation and

use them as guidance to a causal representation. Because the POC is only defined for binary events, we eventually evaluate the POC of the ground truth labels to the predicted labels and use a norm regularizer to encourage the representation function to get rid of spurious features, then we can select a layer of the representation function as our causal representation, and the rest of the functional mapping to the predicted labels naturally acts as a classifier. The experiments demonstrate that our VAE is capable of producing counterfactual examples and thus capable of counterfactual reasoning. As a result, our model outperforms other methods on out-of-distribution classification tasks.

One thing that we need to point out is that our model is not a full generative model like VAE because we have removed the full assumption about the prior of the latent variable and replaced it with a relaxed assumption only involving the support. Since we no longer have the information about the prior, our model is not able to generate artificial data using noise sampled from Gaussian, like VAE, which we have also verified in our experiments. However, this is not an issue for us because our model only conducts counterfactual generation where the latent variables are all inferred from real data. Because of this reason, we regard our model as a discriminative model instead of a generative model.

CHAPTER 4

Conclusion

Ever since the flourishing of deep neural networks, people have been trying to figure out what is really going on inside. Without understanding how the capabilities of NNs are obtained, we can not entrust them with critical tasks, such as autonomous vehicle driving. Additionally, previous works like adversarial attacks have exposed the vulnerability of NNs that we can arbitrarily change the outcome of the network prediction by adding noise messages imperceptible to humans to the input or induce a very confident prediction by NNs using random-looking input data that does not make sense to humans. All of these stem from the fact that NNs are built upon probability or statistics. We know that probability models only learn correlations, it is theoretically impossible for traditional NNs to perform well on out-of-distribution tasks. The ultimate purpose of introducing causality into traditional NNs and more broadly, probabilistic graphical models, is to formulate the distribution shift caused by the change of the underlying causal models and provide methods for countering these o.o.d. changes. We approach this problem from two angles. From the perspective of modularity, when the data distribution shifts, only a few mechanisms in the generative process change. For example, assume we have two datasets, one taken in daylight, and the other taken at night. As long as the object in the image does not change, the only that is different in the two datasets is lightning, the objects in the images are still the same. As a result, to switch from the daylight dataset to the nighttime dataset, we do not need to retrain the model, theoretically, we can only change or fine-tune the mechanism corresponding to the changed mechanism, that is, lighting. Given these modules, we can make the model adapt to a new dataset by intervening and modifying the specific mechanism that reflects the data distribution shift. However, the manipulations mentioned above rely on the premise that we know the specific structure of modules in a NN. In Chapter 2, we propose a framework called competitive disentanglement

to discover modular structures in NNs. This can be considered a top-down approach where we identify high-level modules in NNs. From the perspective of representation learning, we can try to learn a representation that only takes causally stable information, that is, only takes features that truly determine the identity of the data instead of features that have spurious correlations. Such a representation has the potential to truly assess the causal relationships between the predicted label and current features and thus is less prone to misclassification in an out-of-distribution dataset. In Chapter 3, we propose a discriminative VAE that is capable of counterfactual reasoning, and we use it to evaluate the probability of causation which is a measurement that only takes truly causally related features into consideration, and use it to guide the learning of a causal representation. This can be considered a bottom-up approach where we focus on how to make each neuron to causally contribute to the final prediction and reject all spurious features.

Causality has helped us design our methods, and we make use of its power of reasoning to interpret NNs in a way that is consistent with further counterfactuals.

4.1 Chapter Summary and Future Directions

In Chapter 2, we proposed a framework to discover modular structures in causal structural models. The motivation of this work is to discover self-learned modular structures in NNs or NN-like models. By NN-like, we refer to any complicated functional mapping that is a multitude of elementary computing units, the quasi-SCM in our work fits perfectly into this category as it also consists of "neurons" as each node is calculated by a simple function from its parents. From previous works, we know that different tasks may share some common functions inside NNs, which is why fine-tuning a NN to a new domain is a lot easier than training a NN from scratch because there are some modules that are already trained and you only need to fine-tune the part that has to adapt to the new data. According to the Independent Causal Mechanism Principle, if we change one module inside the NN, the functioning of other modules will not be affected. Assuming we have two models working on two datasets, ideally, if we find two substructures inside these two NNs such that switching them (by switching

their parameters) will not change the function or the performance of the two models, then we say the substructures are part of the common mechanisms shared by the two tasks. If we exhaust all possible substructures and find the largest substructures that switching them does not affect performance, then we can say the two maximum substructures are the common mechanism shared by the two models. However, because of the layer-based structure of NNs, it is impossible to iterate over all possible substructures. Instead, we convert the NNs into a class of models where the structural relationships are encoded by an adjacency matrix, which we call quasi-SCMs. Because the optimizer can freely adjust the graphical structure of the model by changing the adjacency matrix, we do not need to worry about which nodes should be selected at the beginning to share the parameters. Any selection would be equivalent because the model can assign these nodes to positions best suited for them. However, a severe drawback of this modification is that the graph will inevitably turn cyclic during training. Thus we use a DAG loss to pull the model back to the domain of acyclic graphs. Eventually, we are able to identify a precise substructure of the NN that is shared by the two tasks which are completely learned by itself without any supervision or structural implications.

As we have discussed in the final section in Chapter 2, we do not to stop at these high-level modules. By carefully designing different tasks we can conduct disentanglement of NN on them, and it is possible to break the structures further down. Also because the intervention is only applied to the high-level modules, structures inside these modules are still learning from a single distribution throughout the whole process. As a result, they may learn spurious causal relationships among neurons within the modules. If we combine causal discovery methods with our mechanism identification method, it is possible to discover modules that are more causally accurate. Also, the modular structures we acquired are in their most primitive form, that is, a set of neurons, so it is hard to directly analyze a single mechanism alone as we have to combine these modules into a complete NN to conduct performance evaluation. If, in the future, we can introduce some standardization procedure to make these modules off-the-shelf, then not only can we directly evaluate them, but we can also add them to other NNs to directly grant them new abilities.

In Chapter 3, we proposed a new VAE model that is capable of producing meaningful counterfactual data and used the probability of causation to formulate a goal based on counterfactual questions to guide the learning of the causal representation. The motivation of this work is to learn causal representations that are resistant to out-of-distribution changes. In other words, we want to eliminate features that have spurious correlations with the labels. The first issue is that we have to define what kind of features should be considered spurious. We argue that only features that appear on all images of different labels but have a stronger correlation with certain labels should be considered as potential spurious features. For example, in the colored MNIST case, if 95% of 3s are blue and 5% are red, and 95% of 8s are red and 5% are blue, then we consider the feature color spurious as all colors appear for all labels, while there is a strong correlation present. However, if we take this to the extreme, and make 100% of 3s blue and 100% 8s red, then we should not consider color as a bad feature because the model does not possess the prior knowledge that we have in order to know what aspects account towards the identities of the numbers. Our goal is to look for features that can consistently indicate the identity of the number. Blue, in this case, consistently appears only for 3, it is just like the actual shape of the number 3 that can consistently define what 3 is. If we do not explicitly tell the model, there is no logical reason for the model to doubt if the feature color is any different from other more sophisticated features, like shape. Similarly, red should also be considered a good feature for identifying the number 8 as red appears if and only if the number is 8. To use the power of reasoning from causal models to discover spurious features, we use the probability of causation to define how a good representation should behave. More specifically, if our representation function captures color as a representation and uses it to infer the predicted label. Then we want to ask the model counterfactual questions like "Given the color is blue, and the input image is a blue 3, the predicted label is also 3, what would the predicted label going to be, had the image been a blue 8?" If color is truly chosen as the representation, then the answer to this question would be that the predicted label would still be 3 as the color would still be blue after the intervention, and this is how we discover spurious features through causal reasoning via counterfactual questions. This question represents a counterfactual thinking process where we depart from reality as we intervene on the label to forcefully set it to 3, but preserve information that is not involved in our intervention. So the

identity of the counterfactual image should change, but all other features like color, distortion, etc., should stay. We need a model that can conduct this counterfactual reasoning. First, we take a careful look at the traditional variational autoencoder model, we found that the traditional assumption that the posterior should be a standard multivariate Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is a huge obstacle for our model to conduct meaningful counterfactual reasoning. The initial intention of this assumption is to increase the power of generation for the VAE model in the sense that the latent variable sampled from the standard Gaussian would be very close to latent variables inferred from realistic data, and consequently, the generated data from the VAE would also be similar to the real data. However, such universal distributions contradict the idea of counterfactuals. Assuming we found the single latent variable directly responsible for the color of the image. Because the color distributions under different labels are different, the NN is bound to learn a conditional distribution generation mechanism inside, which means that the same latent variable may be projected to different colors under different labels. As a result, if given a blue 3 to a vanilla CVAE and we intervene to set the label to 8, then the model is most likely going to produce a red 8 instead of a blue 8 and thus can pass through our counterfactual questions, which reaffirms the theory that the color is a good representation for the number. Such counterfactual is meaningless because we are not generating new data, instead, we are simply sampling from the original distribution where the majority of them are blue 3s and red 8s. From the perspective of disentanglement, our desired disentanglement is not based on probability independence where the label is independent of other latent variables. On the contrary, the disentanglement should be on the level of semantics. In other words, the color-controlling variable may as well be dependent on the label, but what we really want is for the color-controlling variable to represent the same color no matter which label is given. To make the VAE learn such a semantic consistent latent space so that it can conduct meaningful counterfactual reasoning, we first relax the regularization imposed on the posterior and add a counterfactual image generation goal to encourage the VAE to form a consistent semantic latent space. After this, we found that our model is capable of conducting correct counterfactual image generation, where features that are even strongly correlated with the label will be preserved after the label is intervened. Finally, we use the probability of causation to construct a counterfactual representation learning goal to prevent spurious features from

contributing to the final label prediction. Representations learned under such a goal will not contain spurious correlated features and will perform better on o.o.d. samples, which is verified by our experiments.

An issue with this framework is that after we removed the full assumption on the posterior, we do not have a theoretical guarantee for the behavior of the latent space. Although we add a counterfactual generation task to encourage the formation of the consistent semantic latent space, and verified by the experiments, we do not know how well this scales to large NN models. Although we have shown that the regularization should not depend on the densities of the distributions, so we do not think a better probability-based regularization is possible, but we do think a better counterfactual goal may exist which can, to some degree, guarantee a consistent semantic space. Another direction is to further utilize the counterfactual generation ability of our discriminative VAE. As we have shown, our VAE is capable of producing artificial images that are rarely distributed in the original dataset, we can use this model to augment unbalanced datasets by creating more data for those labels that do not have sufficient samples.

A promising direction that combines the power of the two methods we proposed may be the estimation of a theoretical upper bound of the performance of the representations. For the selection of the representation in f_ψ, we still use the traditional representation selection method for NNs, that is, depending on how general you want the representation to be, you can select a layer of the NN represented by f_ψ as the final representation. We believe it is possible to use our quasi-SCM framework to provide a more rational and theoretical ground for choosing the representation. More specifically, if we have two causal representation learning frameworks on two datasets, then we can conduct competitive disentanglement on the two representation learning functions, the common part shared by the two representation functions is supposed to be the representations that work equally well for the two tasks. If we want to get a representation that has a theoretical maximal performance for the two tasks simultaneously, then it should be formed by neurons inside the common part.

4.2 Summary

This thesis has shown that it is possible to use causality to improve the interpretability of NNs and improve their adaptation ability to out-of-distribution data either by directly interpreting the substructures of NNs or by constructing a generative model that is capable of generating counterfactual data so that we can conduct causal reasoning based on the model to improve the causal relevance between neurons in the NN. The modular discovery method can be regarded as a top-down method where we discuss the causal properties of high-level modules, while the causal representation method can be regarded as a bottom-up method where we talk about the causal relationships between neurons. We believe that introducing causal concepts will eventually help us understand more about NNs and maybe one day produces a full interpretation of NNs that is human-readable and guide us to modify NNs to make them act more reliably and reasonably.

Bibliography

[1] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[2] J. Benitez, J. Castro, and I. Requena. Are artificial neural networks black boxes? *IEEE Transactions on Neural Networks*, 8(5):1156–1164, 1997. doi: 10.1109/72.623216.

[3] M. Besserve, A. Mehrjou, R. Sun, and B. Schölkopf. Counterfactuals uncover the modular structure of deep generative models. *arXiv preprint arXiv:1812.03253*, 2018.

[4] M. Besserve, R. Sun, D. Janzing, and B. Schölkopf. A theory of independent mechanisms for extrapolation in generative models. *arXiv preprint arXiv:2004.00184*, 2020.

[5] S. Bongers, P. Forré , J. Peters, and J. M. Mooij. Foundations of structural causal models with cycles and latent variables. *The Annals of Statistics*, 49(5), oct 2021. doi: 10.1214/21-aos2064.

[6] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha. Deep learning for classical japanese literature. *arXiv preprint arXiv:1812.01718*, 2018.

[7] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[8] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis, 2021.

[9] A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/c5ff2543b53f4cc0ad3819a36752467b-Paper.pdf.

[10] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685.

PMLR, 2019.

[11] F.-L. Fan, J. Xiong, M. Li, and G. Wang. On interpretability of artificial neural networks: A survey. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 5(6):741–760, 2021.

[12] P. Forr'e and J. M. Mooij. Markov properties for graphical models with cycles and latent variables. *arXiv: Statistics Theory*, 2017.

[13] P. Forré and J. M. Mooij. Causal calculus in the presence of cycles, latent confounders and selection bias, 2019.

[14] K. Goebel and W. A. Kirk. *Banach's Contraction Principle*, page 7–26. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1990. doi: 10.1017/CBO9780511526152.003.

[15] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2066–2073. IEEE, 2012.

[16] A. Goyal, A. Lamb, J. Hoffmann, S. Sodhani, S. Levine, Y. Bengio, and B. Schölkopf. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019.

[17] M. R. Hestenes. Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5):303–320, 1969.

[18] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[19] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. doi: 10.1038/s41586-021-03819-2. URL https://doi.org/10.1038/s41586-021-03819-2.

[20] N. R. Ke, O. Bilaniuk, A. Goyal, S. Bauer, H. Larochelle, B. Schölkopf, M. C. Mozer, C. Pal, and Y. Bengio. Learning neural causal models from unknown interventions.

arXiv preprint arXiv:1910.01075, 2019.

[21] L. Kirsch, J. Kunze, and D. Barber. Modular networks: Learning to decompose neural computation. *Advances in neural information processing systems*, 31, 2018.

[22] P. Koniusz, Y. Tas, and F. Porikli. Domain adaptation by mixture of alignments of second- or higher-order scatter tensors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4478–4487, 2017.

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

[24] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[25] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[26] C. Lu, Y. Wu, J. M. Hernández-Lobato, and B. Schölkopf. Nonlinear invariant risk minimization: A causal approach. *arXiv preprint arXiv:2102.12353*, 2021.

[27] M. Makar, B. Packer, D. Moldovan, D. Blalock, Y. Halpern, and A. D'Amour. Causally motivated shortcut removal using auxiliary labels. In *International Conference on Artificial Intelligence and Statistics*, pages 739–766. PMLR, 2022.

[28] G. Montavon, W. Samek, and K.-R. Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

[29] J. M. Mooij and T. Claassen. Constraint-based causal discovery using partial ancestral graphs in the presence of cycles. In *UAI*, 2020.

[30] J. M. Mooij and T. Heskes. Cyclic causal discovery from continuous equilibrium data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, page 431–439, Arlington, Virginia, USA, 2013. AUAI Press.

[31] J. M. Mooij, D. Janzing, T. Heskes, and B. Schölkopf. On causal discovery with cyclic additive noise models. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[32] J. M. Mooij, S. Magliacane, and T. Claassen. Joint causal inference from multiple contexts. *J. Mach. Learn. Res.*, 21(1), jun 2022. ISSN 1532-4435.

[33] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*. IEEE, 2010.

[34] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[35] G. Parascandolo, N. Kilbertus, M. Rojas-Carulla, and B. Schölkopf. Learning independent causal mechanisms. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4036–4044. PMLR, 10–15 Jul 2018.

[36] J. Pearl. *Causality*. Cambridge university press, 2009.

[37] J. Pearl et al. Models, reasoning and inference. *Cambridge, UK: CambridgeUniversityPress*, 19:2, 2000.

[38] J. Peters, P. Bühlmann, and N. Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):947–1012, 2016.

[39] J. Peters, D. Janzing, and B. Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[40] G. Plumb, M. Al-Shedivat, A. A. Cabrera, A. Perer, E. Xing, and A. Talwalkar. Regularizing black-box models for improved interpretability. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 10526–10536. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/770f8e448d07586afbf77bb59f698587-Paper.pdf.

[41] A. Puli, L. H. Zhang, E. K. Oermann, and R. Ranganath. Predictive modeling in the presence of nuisance-induced spurious correlations. *arXiv preprint arXiv:2107.00520*, 2021.

[42] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[43] E. Rosenfeld, P. Ravikumar, and A. Risteski. The risks of invariant risk minimization. *arXiv preprint arXiv:2010.05761*, 2020.

[44] S. Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[45] D. S. Sachan and G. Neubig. Parameter sharing methods for multilingual self-attentional translation models. *arXiv preprint arXiv:1809.00252*, 2018.

[46] A. Sauer and A. Geiger. Counterfactual generative networks. *arXiv preprint arXiv:2101.06046*, 2021.

[47] B. Schölkopf. *Causality for Machine Learning*, page 765–804. Association for Computing Machinery, New York, NY, USA, 1 edition, 2022. ISBN 9781450395861. URL https://doi.org/10.1145/3501714.3501755.

[48] B. Schölkopf. Causality for machine learning. In *Probabilistic and Causal Inference: The Works of Judea Pearl*, pages 765–804. 2022.

[49] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-cam: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016.

[50] R. Suter, D. Miladinovic, B. Schölkopf, and S. Bauer. Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness. In *International Conference on Machine Learning*, pages 6056–6065. PMLR, 2019.

[51] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[52] T. Teshima, I. Sato, and M. Sugiyama. Few-shot domain adaptation by causal mechanism transfer. In *International Conference on Machine Learning*, pages 9458–9469. PMLR, 2020.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[54] V. Veitch, A. D'Amour, S. Yadlowsky, and J. Eisenstein. Counterfactual invariance to spurious correlations: Why and how to pass stress tests. *arXiv preprint arXiv:2106.00545*, 2021.

[55] S. Wachter, B. D. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Cybersecurity*, 2017.

[56] W. Wang, X. Lin, F. Feng, X. He, M. Lin, and T.-S. Chua. Causal representation learning for out-of-distribution recommendation. In *Proceedings of the ACM Web Conference 2022*, pages 3562–3571, 2022.

[57] Y. Wang and M. I. Jordan. Desiderata for representation learning: A causal perspective. *arXiv preprint arXiv:2109.03795*, 2021.

[58] M. A. Wiering. Evolving causal neural networks. In *Benelearn'02: Proceedings of the Twelfth Belgian-Dutch Conference on Machine Learning*, pages 103–108, 2002.

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

[60] M. Yang, F. Liu, Z. Chen, X. Shen, J. Hao, and J. Wang. Causalvae: Disentangled representation learning via neural structural causal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9593–9602, 2021.

[61] Z. Yue, Q. Sun, X.-S. Hua, and H. Zhang. Transporting causal mechanisms for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8599–8608, 2021.

[62] Q. Zhang, Y. N. Wu, and S.-C. Zhu. Interpretable convolutional neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2018. doi: 10.1109/CVPR.2018.00920.

[63] Q. Zhang, Y. Gu, M. Mateusz, M. Baktashmotlagh, and A. Eriksson. Implicitly defined layers in neural networks, 2020.

[64] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742, 2021. doi: 10.1109/TETCI.2021.3100641.

[65] X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing. Dags with no tears: Continuous optimization for structure learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[66] X. Zheng, C. Dan, B. Aragam, P. Ravikumar, and E. P. Xing. Learning sparse nonparametric DAGs. In *International Conference on Artificial Intelligence and Statistics*, 2020.