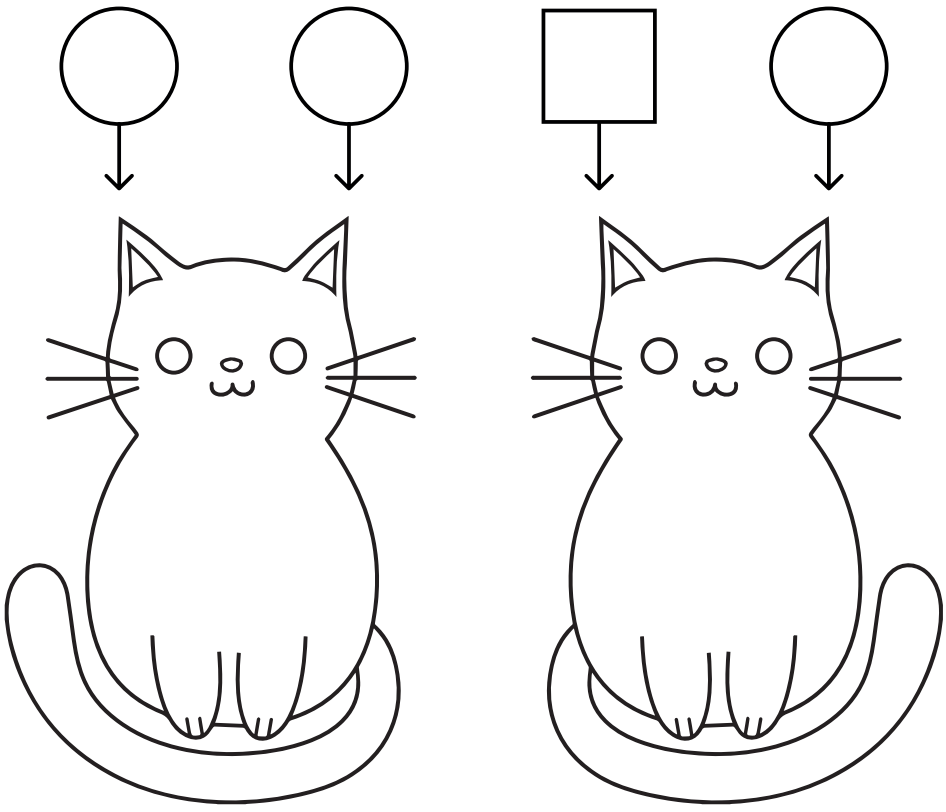
Invariance in deep representations

Ilse, M.

Publication date
2022
Document Version
Final published version

Invariance in Deep Representations



Maximilian Ilse

Invariance in Deep Representations

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. P.P.C.C. Verbeek
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Aula der Universiteit
op vrijdag 14 oktober 2022, te 11.00 uur

door Maximilian Ilse
geboren te Halle Saale

Promotiecommissie

Promotor:	prof. dr. M. Welling	Universiteit van Amsterdam
Copromotor:	dr. J.M. Tomczak	Universiteit van Amsterdam
Overige leden:	prof. dr. J.M. Mooij	Universiteit van Amsterdam
	prof. dr. ir. C.I. Sánchez Gutiérrez	Universiteit van Amsterdam
	prof. dr. I. Išgum	Universiteit van Amsterdam
	dr. S. Magliacane	Universiteit van Amsterdam
	dr. S. Bauer	KTH Stockholm
	prof. dr. O. Winther	Technical University of Denmark

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Preface

In this thesis, Invariance in Deep Representations, we propose novel solutions to the problem of learning invariant representations. We adopt two distinct notions of invariance. One is rooted in symmetry groups and the other in causality. Last, despite being developed independently from each other, we aim to take a first step towards unifying the two notions of invariance.

- We propose a neural network-based permutation-invariant aggregation operator that corresponds to the attention mechanism (Section 2). We develop a novel approach for set classification. Notably, an application of the proposed attention-based operator provides insight into the contribution of each element to the set label.

- We find that application-oriented research areas like medical imaging or robotics, data augmentation techniques are used to learn domain invariant features. We demonstrate that causal concepts can be used to explain the success of data augmentation by describing how they can weaken the spurious correlation between the observed domains and the task labels (Section 3). We demonstrate that data augmentation can serve as a tool for simulating interventional data. We use these theoretical insights to derive a simple algorithm that is able to select data augmentation techniques that will lead to better domain generalization.

- We propose a novel causal reduction method (Section 4) that replaces an arbitrary number of possibly high-dimensional latent confounders with a single latent confounder that lives in the same space as the treatment variable without changing the observational and interventional distributions entailed by the causal model. After the reduction, we parameterize the reduced causal model using a flexible class of transformations, so-called normalizing flows. We propose a learning algorithm to estimate the parameterized reduced model jointly from observational and interventional data.

- We propose the Domain Invariant Variational Autoencoder (Section 5), a generative model that tackles the problem of domain shifts by learning three independent latent subspaces, one for the domain, one for the class, and one for any residual variations. We show that due to the generative nature of our model we can also incorporate unlabeled data from known or previously unseen domains.

Amsterdam, August 25, 2022,

Maximilian Ilse

Samenvatting

In dit proefschrift, Invariance in Deep Representations, stellen we nieuwe oplossingen voor voor het probleem van het leren van invariant representations. We nemen twee verschillende noties van invariantie. De ene is geworteld in symmetry groups en de andere in causality. Tenslotte willen we, ondanks het feit dat ze onafhankelijk van elkaar zijn ontwikkeld, een eerste stap zetten in de richting van het verenigen van de twee noties van invariance.

- Wij stellen een op een neuraal netwerk gebaseerde permutatie-invariante aggregatieoperator voor die overeenstemt met het attention mechanism (Sectie 2). We ontwikkelen een nieuwe aanpak voor set classificatie. In het bijzonder, een toepassing van de voorgestelde attention-based operator geeft inzicht in de bijdrage van elk element aan het set label.

- We vinden dat in toepassingsgerichte onderzoeksgebieden zoals medische beeldvorming of robotica, data augmentation technieken worden gebruikt om domain invariant kenmerken te leren. We tonen aan dat causale concepten kunnen worden gebruikt om het succes van data augmentaion te verklaren door te beschrijven hoe ze de onechte correlatie tussen de waargenomen domain en de task label kunnen verzwakken (Sectie 3). We tonen aan dat data augmentaion kan dienen als een hulpmiddel voor het simuleren van interventional data. We gebruiken deze theoretische inzichten om een eenvoudig algoritme af te leiden dat in staat is om data augmentation technieken te selecteren die zullen leiden tot een betere domain generalization.

- Wij stellen een nieuwe causale reductiemethode voor (Sectie 4) die een willekeurig aantal mogelijk hoog-dimensionale latent confounders vervangt door één enkele latent confounder die in dezelfde space als de behandelingsvariabele leeft zonder de observational en interventional verdelingen te wijzigen die het causale model met zich meebrengt. Na de reductie parametriseren we het gereduceerde causale model met behulp

van een flexibele klasse van transformaties, de zogenaamde normalizing flows. Wij stellen een leeralgoritme voor om het geparametriseerde gereduceerde model gezamenlijk uit observational en interventional data te trainen.

- Wij stellen de Domain Invariant Variational Autoencoder voor (Sectie 5), een generatief model dat het probleem van domain generalization aanpakt door drie onafhankelijke latent subspaces te leren, één voor het domain, één voor de task, en één voor eventuele residuele variaties. We laten zien dat we door de generatieve aard van ons model ook ongelabelde data van bekende of voorheen ongeziene domains kunnen opnemen.

Contents

List of Publications

This thesis consists of an introduction, a conclusion, and of the following publications:

I Maximilian Ilse, Jakub M. Tomczak, and Max Welling. Attention-based Deep Multiple Instance Learning. In *Thirty-fifth International Conference on Machine Learning*, 2018.

II Maximilian Ilse, Jakub M. Tomczak, and Patrick Forré. Selecting Data Augmentation for Simulating Interventions . In *Thirty-eighth International Conference on Machine Learning*, 2021.

III Maximilian Ilse, Patrick Forré, Max Welling, and Joris M. Mooij. Efficient Causal Inference from Combined Observational and Interventional Data through Causal Reductions. In *Under submission*, 2021.

IV Maximilian Ilse, Jakub M. Tomczak, Christos Louizos, and Max Welling. DIVA: Domain Invariant Variational Autoencoders. In *Medical Imaging with Deep Learning*, 2020.

Author's Contribution

Publication I: "Attention-based Deep Multiple Instance Learning"

- M. Ilse: text, figures, and experiments
- J. M. Tomczak: text, experiments, and supervision
- M. Welling: text, and supervision

Publication II: "Selecting Data Augmentation for Simulating Interventions "

- M. Ilse: text, figures, and experiments
- J. M. Tomczak: text, and supervision
- P. Forré: text, and supervision

Publication III: "Efficient Causal Inference from Combined Observational and Interventional Data through Causal Reductions"

- M. Ilse: text, figures, and experiments
- P. Forré: text, and supervision

Author's Contribution

- M. Welling: text, and supervision
- J. M. Mooij: text, experiments, and supervision

Publication IV: "DIVA: Domain Invariant Variational Autoencoders"

- M. Ilse: text, figures, and experiments
- J. M. Tomczak: text, and supervision
- C. Louizos: text, and supervision
- M. Welling: text, and supervision

List of Figures

List of Tables

Abbreviations

AI Artificial Intelligence

AUC Area Under the receiver operating characteristic Curve

CNN Convolutional Neural Network

CDANN Conditional Domain Adversarial Neural Network

CVIB Conditional Variational Information Bottleneck

DAG Directed Acyclic Graph

DANN or DA Domain Adversarial Neural Network

DIVA Domain Invariant Variational Autoencoder

DNN Deep Neural Network

DSR Disentangled Semantic Representations

Elbo Evidence lower bound

ERM Empirical Risk Minimization

GAN Generative Adversarial Network

G-CNN Group equivariant Convolutional Neural Network

ICP Invariant Causal Prediction

IRM Invariant Risk Minimization

H&E Hematoxylin and Eosin

MIL Multiple Instance Learning

MSE Mean squared Error

ReLU REctified Linear Unit

ROI Region Of Interest

SCM Structural Causal Model

SDA Select Data Augmentation

SVM Support Vector Machine

VAE Variational AutoEncoder

Notation

x a scalar

$\mathbf{x}$ a vector

X a scalar valued random variable or a set or a matrix

$\mathbf{X}$ a vector valued random variable

$p(X = x)/p(\mathbf{X} = \mathbf{x})$ a probability density function, where we use $p(x)/p(\mathbf{x})$ short for $p(X = x)/p(\mathbf{X} = \mathbf{x})$

$P(X \leq x) = \int_{-\infty}^{x} p(x')dx'$ a cumulative distribution function

$\mathscr{X}$ a space

$\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{\omega}$ trainable parameters

1. Introduction and Background

1.1 The importance of invariance

In the last twenty years, we have seen an enormous jump in the performance of Artificial Intelligence (AI) systems driven by deep learning. Among other things, these systems can recognize objects in images or videos, generate entire pages of text, and predict the chemical properties of molecules. As a result, we will encounter more and more AI systems in our everyday lives, be it the autocomplete in messaging apps, the driving assistant in cars, or the diagnostic tools in hospitals. The latter two use cases are prime examples of safety-critical applications, where mistakes of the AI system can lead to fatalities in the worst case. This poses the important question:

"Are modern AI systems reliable and safe to use?"

An ever-growing collection of research shows how sensitive deep learning-based AI models can be to small input changes. For example, a slight change in the lighting condition, color, or perspective of an object can lead to a drastic decrease in the performance of an AI system [Azulay and Weiss, 2019].

In contrast to AI systems, humans excel at adapting to new environments. One might have never seen a red banana; still, nobody would mistake it for a tomato when first encountered. We, as humans, are capable of separating the color from the shape of an object and subsequently only rely on the shape information for our assessment. In another instance, we can separate the shape of an object from its orientation. We are therefore able to recognize another human even if they stand on their head. In both cases, we as humans ignore parts of the visual input. More formally, we can describe the predictions we make as *invariant* to the color of the banana or the orientation of the human.

A safe and reliable AI system would learn to be robust to changes in the visual input in the same way a human is. While the examples above focus on vision tasks, there are various modalities where invariance is crucial. Suppose we want to make predictions about physical or chemical systems. In that case, the AI system needs to capture the underlying symmetries, where each symmetry leads to a conserved quantity as described by Noether's theorem [Noether and Tavel, 1971]. Furthermore, in healthcare applications, we want to make sure that AI systems generalize across all subpopulations encountered during deployment. In many cases, it will be required to be invariant to changes in covariates like age, sex, and ethnicity.

In the present thesis, we will focus on two distinct notions of invariance. One is rooted in symmetry groups and the other in causality. We will give numerous examples for each of the invariances and enforce them using deep learning. Last, despite being developed independently from each other, we aim to take a first step towards unifying the two notions of invariance.

1.2 Two notions of invariance

In this section, we will introduce two notions of invariance. The first notion of invariance is derived from the concept of group symmetries. In contrast, the second notion is based on the concept of interventions originating from causality literature. We will see that the two notions lead to a variety of approaches for training machine learning models. Yet, disregarding the approach chosen, the result is an invariant machine learning model, i.e., some changes in the input of a machine learning model will not change its output.

1.2.1 Symmetry groups

Symmetry is the property of a mathematical object to remain unchanged under a set of transformations. Therefore symmetry is considered a type of invariance. Arguably, the most familiar example of symmetry is the symmetry of shapes. If we consider the equilateral triangle in Figure 1.1 we find that it has a variety of symmetries. For example, if we rotate the triangle by 120°, 240°, or 360° angles, Figure 1.1 would still look the same. Another type of symmetry that can be found in Figure 1.1 is reflection symmetry. There are three different mirror lines through which we can reflect the triangle without changing its appearance.

In order to formalize the notion of symmetries we are going to use the mathematical concept of groups. Consider a set G and a binary operator denoted as "$\cdot$". G and "$\cdot$" form a group if they satisfy the following conditions,

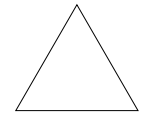


Figure 1.1. Equilateral triangle.

also known as group axioms:

- (closure) G is closed under "$\cdot$": if $g,h \in G \Rightarrow g \cdot h \in G$.
- (identity element) There exists an identity element $i \in G$: For all $g \in G$ we have $g \cdot i = i \cdot g = g$.
- (inverse element) Every element $g \in G$ has an inverse in G: For all $g \in G$ there exists an element $g^{-1} \in G$ such that $g \cdot g^{-1} = g^{-1} \cdot g = i$.
- (associativity) The binary operator "$\cdot$" has the associative property: for all $g,h,j \in G$ we have $g \cdot (h \cdot j) = (g \cdot h) \cdot j$.

Furthermore, we call a function that maps from $G \times \mathscr{X}$ to $\mathscr{X}$ a group action and use the short notation $g \cdot \mathbf{x}$ for $g \in G$ and $\mathbf{x} \in \mathscr{X}$. Last, if a set and a binary operator only satisfy the conditions of closure and associativity, it is called a semigroup.

One common example of a symmetry group is the cyclic group C_n. The group consists of all rotations around a fixed point by multiples of the angle $360°/n$. For the triangle in Figure 1.1 the associated symmetry group is $C_3 = \{120°, 240°, 360°\}$. If we take the three additional reflections into account, the associated group is the dihedral group D_3.

However, shapes are not the only mathematical objects that can have a symmetry. Consider the following common machine learning task. We have a set of natural images [1] $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, $\mathbf{x}_i \in \mathscr{X}$, where $\mathscr{X}$ is the input space. In addition, we have a set of corresponding binary labels $Y = \{y_1, ..., y_N\}$, $y_i \in \mathscr{Y} = \{0,1\}$. Last we assume that there exists a labeler function $f : \mathscr{X} \to \mathscr{Y}$, that maps the images to their respective class.

In this particular example, we know that there exists a set of transformations that does not change the label y_i of an image $\mathbf{x}_i$. Among these transformations are translations, rotations, and reflections. In Figure 1.2, we show an image of a cat and its horizontal reflection. We know that horizontal reflections will not change the label of the image. If we consider the group of horizontal reflections G and the labeler function f, the function f is an invariant map if

$$f(g \cdot \mathbf{x}) = f(\mathbf{x}) \forall\, g \in G. \tag{1.1}$$

[1] Natural images are photos of everyday objects like planes, cars and cats.

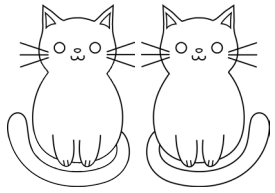


Figure 1.2. Mirrored cat.

In this example, the labeler function f is invariant with respect to the group of horizontal reflections.

In the following, we will summarize the two most common approaches found in the deep learning literature to enforce group symmetries. In both cases, the symmetry of a dataset and the associated task needs to be known apriori.

Group-equivariant neural networks The research field of group-equivariant neural networks focuses on designing neural networks that are invariant or equivariant with respect to a specific symmetry group, where a function f is an equivariant map if it commutes with a group action $g \in G$, i.e.,

$$f(g \cdot \mathbf{x}) = g \cdot f(\mathbf{x}) \forall g \in G. \tag{1.2}$$

In Cohen and Welling [2016], the Group equivariant Convolutional Neural Network (G-CNN) was introduced. While the convolutional weight sharing in standard Convolutional Neural Networks (CNNs) gives rise to equivariance with respect to the group of translations, G-CNNs generalize this concept to a larger group of symmetries. By replacing convolutional layers with group-convolutional layers, Cohen and Welling [2016] were able to build neural networks that are invariant to rotations and reflections of the inputs. Unfortunately, group-equivariant neural networks are restricted to transformations that can be expressed as a group, such as translation, rotation, and reflection. However, many real-world transformations cannot be expressed as a group, for example, scaling [2], occlusion, and the saturation of colors.

Data augmentation Data augmentation describes a loose collection of transformations of the input data $\mathbf{x}$. The transformed input data is added to the training set to make the model invariant beyond the symmetries build into the architecture, e.g., translation symmetry in the case of CNNs. Common data augmentation techniques for image data are horizontal/vertical flipping, rotation, occlusion, color augmentation, and scaling. The advantage of data augmentation is that these techniques are not limited to transformations that form a group. For example, the scaling of a discretized image does not always have an inverse transformation. The set

[2] If applied to discretized images

of scaling transformations on the space of discretized images is therefore considered a semigroup. As a result, data augmentation can be used to enforce invariance to a wider variety of transformations of the input space. However, while the group-equivariant neural networks described in the previous section enforce strict equivariance or invariance, data augmentation only leads to approximate invariance. Approximate invariance [Chen et al., 2020a] is defined as follows:

$$\mathbf{x} \approx_d g \cdot \mathbf{x}, \tag{1.3}$$

where $\mathbf{x} \in \mathscr{X}$ and $\approx_d$ denotes approximate equality in distribution. I.e., the data distribution does not change too much under a group action. There exist a plethora of data augmentation techniques for image, audio, and text data. However, for other modalities like gene sequences, there exist no established data augmentation techniques. Last, in most cases, expert knowledge is needed to manually choose the data augmentation technique that enforces the correct symmetry.

1.2.2 Invariance under intervention

In Section 1.2.1, we developed a notion of invariance using the concept of symmetry groups. However, a second notion of invariance was developed independently, not relying on group theory. The core idea is that a ground-truth causal model is responsible for generating the data we observe. As such, the causal model captures the relationship between all variables in a system. We make use of the assumption that the causal model consists of several independent causal mechanisms. As a result, some components remain unchanged when altering a subset of the variables in a causal model. These components are called *invariant*. We will now introduce a minimum set of concepts from the causality literature that will enable us to formally define the notion of invariance from a causal point of view.

Graphical models and SCMs Causal models represent the underlying mechanism by which the data we observe was generated. A powerful tool for visualizing and reasoning about causal models are so-called graphical models. Each variable of a causal model is represented by a node and each mechanism between variables by an edge. In Figure 1.3, we see a graphical model that consists of three random variables $\mathbf{X}, Y$, and $\mathbf{Z}$ and three directed edges $\mathbf{X} \to Y$, $\mathbf{Z} \to \mathbf{X}$, $\mathbf{Z} \to Y$. In Figure 1.3 there is an arrow from $\mathbf{Z}$ to $\mathbf{X}$ and to Y, we therefore call $\mathbf{Z}$ a common cause of $\mathbf{X}$ and Y or a *confounder*. In general, if we do not control for $\mathbf{Z}$, e.g., by conditioning, our estimate of the causal effect of $\mathbf{X}$ on Y will be biased.

In the following, we will only focus on so-called Directed Acyclic Graphs (DAGs). DAGs have two important properties: (i) all edges have a distinct

direction, indicating causation. (ii) there are no cycles in a graph. The graph shown in Figure 1.3 is a DAG.

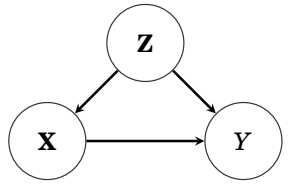


Figure 1.3. Example DAG.

Every DAG entails a factorization of the joint distribution of all variables in the graph. For the DAG in Figure 1.3 the factorization is $p(\mathbf{x},y,\mathbf{z}) = p(y|\mathbf{x},\mathbf{z})p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$. In order to make general statements for a set of DAGs we will need the concept of parent and children nodes. In the DAG in Figure 1.3 we call the set $pa_Y = \{\mathbf{x},\mathbf{z}\}$ the parents of Y. The concept of parent nodes lets us write the factorization of any DAG as

$$p(\mathbf{x}_1,\dots,\mathbf{x}_K) = \prod_{j=1}^{K} p(\mathbf{x}_j|pa_{\mathbf{X}_j}). \tag{1.4}$$

Every DAG is associated with a so-called Structural Causal Model (SCM). An SCM is a collection of variables and deterministic functions that describe all the causal mechanisms of the data generating process. The SCM that corresponds to the DAG in Figure 1.3 is the following

$$\begin{aligned}\mathbf{Z} &= \mathbf{N}_\mathbf{Z}\\ \mathbf{X} &= f(\mathbf{Z},\mathbf{N}_\mathbf{X})\\ Y &= g(\mathbf{X},\mathbf{Z},\mathbf{N}_Y),\end{aligned} \tag{1.5}$$

where $\mathbf{N}_\mathbf{X},\mathbf{N}_Y,\mathbf{N}_\mathbf{Z}$ are sampled from the jointly independent noise distributions $p(\mathbf{N}_\mathbf{X})$, $p(\mathbf{N}_Y)$, $\mathrm{p}(\mathbf{N}_\mathbf{Z})$. In the next section, we will see that SCMs enable us to compute interventional and counterfactual distributions.

Interventions In Figure 1.4 the DAG after an intervention on $\mathbf{X}$ is shown. In contrast to the DAG in Figure 1.3, $\mathbf{X}$ does no longer depend on $\mathbf{Z}$ as indicated by the missing arrow between $\mathbf{X}$ and $\mathbf{Z}$.

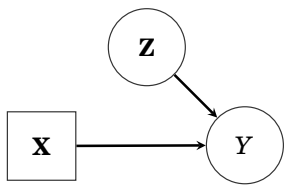


Figure 1.4. Example DAG after intervention on $\mathbf{X}$.

The distribution entailed by the DAG in 1.4 is called the interventional distribution. In general the interventional distribution will differ from the distribution entailed by the original DAG in Figure 1.3. Formally we write

$$p(y,\mathbf{z}|\mathrm{do}(\mathbf{x})) \neq p(y,\mathbf{z}|\mathbf{x}), \tag{1.6}$$

where $p(y,\mathbf{z}|\mathrm{do}(\mathbf{x}))$ is the interventional distribution and $p(y,\mathbf{z}|\mathbf{x})$ is the conditional distribution. We use the do-*operator* [Pearl, 2009] to indicate when we perform an intervention.

Interventions let us reason about the effect on a system if we alter the state of one or more variables. Consider a random variable $\mathbf{X}_j$, its set of parents $pa_{\mathbf{X}_j}$, and the functional assignment $\mathbf{X}_j = f(pa_{\mathbf{X}_j},\mathbf{N}_{\mathbf{X}_j})$. We define two different types of interventions on $\mathbf{X}_j$:

- Hard intervention: $\mathrm{do}(\mathbf{x}_j = \mathbf{a})$
- Noise intervention: $\mathrm{do}(\mathbf{x}_j = \widetilde{N}_{\mathbf{X}_j})$, where $\widetilde{N}_{\mathbf{X}_j} \sim p(\widetilde{N}_{\mathbf{X}_j})$, $p(\widetilde{N}_{\mathbf{X}_j})$ being an independent noise distribution

In both cases, $\mathbf{X}_j$ becomes independent of its parents. Graphically speaking, an intervention is removing all incoming arrows, as seen in Figure 1.4. Last, we revisit the SCM in Equation 1.5. If we were to perform a noise intervention on $\mathbf{X}$, the SCM after intervention is

$$\begin{aligned}\mathbf{Z} &= \mathbf{N}_\mathbf{Z}\\ \mathbf{X} &= \widetilde{N}_\mathbf{X}\\ Y &= g(\mathbf{X},\mathbf{Z},\mathbf{N}_Y).\end{aligned} \tag{1.7}$$

We see that the value of $\mathbf{X}$ is independent of $\mathbf{Z}$.

Independent Mechanisms After having defined the necessary concepts, we can now introduce a causal notion of invariance. In the following, we will closely follow the example in Peters et al. [2017]. We start with two random variables that describe two properties of a city: altitude A and average annual temperature T. From a purely statistical point of view, there are two ways of factorizing the joint probability $p(a,t)$

$$p(a,t) = p(a|t)p(t) \tag{1.8}$$

$$= p(t|a)p(a), \tag{1.9}$$

where none is preferred over the other. From a causal perspective $p(a,t) = p(a|t)p(t)$ corresponds to the causal graph $T \to A$, i.e., temperature causes altitude, whereas $p(a,t) = p(t|a)p(a)$ corresponds to the causal graph $A \to T$, i.e., altitude causes temperature. We now ask the question: "Which is the correct causal graph?", where 'correct' is defined as the graph that generated the data we observe in the first place.

We can test the plausibility of the two factorizations by thinking about the effect of interventions on A and T would have. If we were to change the altitude of a city hypothetically, we would expect the average annual temperature to change. In contrast, common sense tells us that changing the average annual temperature of a city would not change its altitude. In summary, intervening on A causes T to change, but intervening on T does not change A. We find that the correct causal graph is $A \rightarrow T$, which entails the factorization $p(a,t) = p(t|a)p(a)$.

Furthermore, we can consider the joint distribution of average annual temperature and altitude for two different countries: The Netherlands $p^N(a,t)$ and Chile $p^C(a,t)$. As shown above, the causal graph $A \rightarrow T$ leads to the following factorizations: $p^N(a,t) = p(t|a)p^N(a)$ and $p^C(a,t) = p(t|a)p^C(a)$. While the distributions of the altitude $p(a)$ will vary strongly between the Netherlands and Chile the mechanism $p(t|a)$ is independent of $p(a)$. Therefore $p(t|a)$ is called an independent or invariant mechanism.

Using the definition of a set of parent variables in Section 1.2.2, we can formalize this notion for arbitrary DAGs. Let us consider an SCM $\mathscr{C}$ with random variables $\mathbf{X}_1,\ldots,\mathbf{X}_N$, then

$$p^{\widetilde{\mathscr{C}}}\left(\mathbf{x}_j|pa_{\mathbf{X}_j}\right) = p^{\mathscr{C}}\left(\mathbf{x}_j|pa_{\mathbf{X}_j}\right), \tag{1.10}$$

for any SCM $\widetilde{\mathscr{C}}$ that is constructed from $\mathscr{C}$ by intervening on some $\mathbf{X}_k$ but not on $\mathbf{X}_j$. Equation 1.10 tells us that the mechanism $p(\mathbf{x}_j|pa_{\mathbf{X}_j})$ is invariant to interventions except an intervention on $\mathbf{X}_j$.

Learning invariant causal mechanisms The most reliable way of reducing the confounding bias is obtaining data from a Randomized Controlled Trial (RCT). The goal of an RCT is to remove all confounding biases by randomization. In the SCM in Equation 1.7, we have already seen how a noise intervention can be used to remove confounding. In this example, we replaced the confounded values of $\mathbf{X}$ with random samples from the noise distribution $p(\widetilde{N}_{\mathbf{X}})$. In the following, we will call data generated by a known intervention or an RCT *interventional* or *experimental* data in contrast to observational data that is possibly biased. In general, if we had access to a dataset where all bias is removed, we could directly learn the invariant mechanism. However, in many application domains like healthcare, it is often costly, risky, unethical, or simply impossible to perform an RCT. As a result, in cases where we are able to perform an RCT, the number of samples is usually small.

Therefore, many methods aim to learn invariant mechanisms purely from observational data. Unfortunately, each of those methods relies on additional, mostly untestable, assumptions or the existence of other variables, which limits their practical use [Pearl, 1995, Wang and Blei, 2019]. A detailed discussion of these methods is out of scope for the present thesis. Instead, in Chapter 3 we will show how to simulate interventions

using data augmentation and in Chapter 4 we will show how to combine interventional and observational data without the drawbacks mentioned above.

Last, we will consider combining data from different so-called environments or domains to learn invariant machine learning models [Peters et al., 2016]. We assume a different set of interventions was performed for each domain, i.e., each domain is equal to a different interventional distribution. However, we do not know the exact nature of each intervention nor which variables were changed. We only have access to an additional variable D that indicates from which domain each data point was sampled. For example, for a simple classification task with input $\mathbf{X}$ and target Y from different domains, the training dataset is the set $\{\mathbf{x}_i, y_i, d_i\}_{i=1}^N$. This is a common scenario in healthcare applications where we have access to data from, e.g., different hospitals and populations. In the previous section, we have seen that an invariant model is invariant to interventions on all but the target variable Y. In the case of learning with different environments, we reverse this definition and use it for learning. If we can learn one mechanism that generalizes to all training domains, we have learned the true invariant causal mechanism with high probability.

1.3 Deep generative modeling

In the second half of this thesis, Chapter 4 and 5, we will see that deep generative models are a powerful tool for learning invariant models. By modeling the generative process, they can learn invariant mechanisms that will generalize across datasets. We will focus on two classes of deep generative models in the present thesis: variational autoencoders and normalizing flows. In the following, we will introduce both classes focusing on why they are especially well suited for learning invariant models.

1.3.1 Variational Autoencoder

In 2014, Kingma and Welling [2013] and Rezende et al. [2014] introduced a novel deep latent variable model: the Variational AutoEncoder (VAE). In Figure 1.5, the DAG of an VAE in its simplest form is shown. It consists of only two nodes, the observed data $\mathbf{X}$ and the latent variable $\mathbf{Z}$.

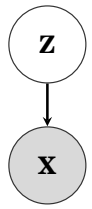


Figure 1.5. Graphical model of an VAE.

First, we show how to approximate the marginal likelihood of the data $p(\mathbf{x})$ by introducing a latent variable $\mathbf{Z}$. Using $\mathbf{Z}$ we can express $p(\mathbf{x})$ as

$$p(\mathbf{x}) = \int p(\mathbf{x},\mathbf{z})d\mathbf{z}. \tag{1.11}$$

Since the integral in Equation 1.11 is in general intractable, the VAE is instead maximizing the so-called Evidence lower bound (Elbo). For a single data point $\mathbf{x}_i$ the bound is given by

$$\log p(\mathbf{x}_i) \geq \mathrm{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}_i)}\left[\log p_{\boldsymbol{\phi}}(\mathbf{x}_i|\mathbf{z})\right] - KL(q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}_i)||p(\mathbf{z})) = Elbo(\boldsymbol{\theta},\boldsymbol{\phi},\mathbf{x}_i). \tag{1.12}$$

The VAE is called a deep generative model since the posterior $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}_i)$ and the likelihood $p_{\boldsymbol{\phi}}(\mathbf{x}_i|\mathbf{z})$ are parameterized by deep neural networks. Furthermore, the prior $p(\mathbf{z})$ plays an important role in distinguishing the VAE from a standard autoencoder. Minimizing the KL divergence from the posterior to the prior regulates the capacity of the latents. In addition, the latent space $\mathcal{Z}$ has a lower dimension than the input space $\mathcal{X}$. Restricting the capacity of $\mathbf{Z}$ encourages that images that are semantically close along one axis are close in latent space. For example, it is beneficial to cluster all red objects using one latent and all rotated objects using another latent. What adds to this pressure is the factorization of the posterior and prior, i.e., $p(\mathbf{z}) = \prod_{i=1}^{D} p(z_i)$, $q(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^{D} q(z_i|\mathbf{x})$. As a result, VAEs naturally tend to learn a *disentangled* representation $\mathbf{Z}$ [Higgins et al., 2018].

For the mirrored cat example in Section 1.2.1, a disentangled representation would look as follows: one latent is used for the orientation of the cat (mirrored, not mirrored), and one latent captures the shape of the cat. Suppose we are interested in a classifier for cats invariant of their orientation. In that case, we can simply ignore the latent used to encode orientation and build a classifier on top of the latent that encodes shape. The model we obtain will be invariant to changes in the orientation of the input.

We now define disentanglement from a perspective of group symmetry as seen in Higgins et al. [2018]. We start with considering the group action $g\cdot\mathbf{x}: G\times\mathcal{X}\to\mathcal{X}$. Furthermore, we assume that the group G decomposes as a direct product $G = G_1\times G_2$. The group action is called disentangled with respect to the decomposition of G if

$$(g_1,g_2)\cdot(\mathbf{x}_1,\mathbf{x}_2) = (g_1\cdot\mathbf{x}_1, g_2\cdot\mathbf{x}_2). \tag{1.13}$$

That is, a group element $g_1\in G_1$ acts on $\mathbf{x}_1$ but leaves $\mathbf{x}_2$ unchanged and vice versa. Now, we apply this definition of disentanglement to a representation $\mathbf{Z}$ learned by an VAE. We consider the set of observations $X = \{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$ and corresponding representations $Z = \{\mathbf{z}_1,\ldots,\mathbf{z}_N\}$, where $\mathbf{z} = f(\mathbf{x})$, $f:\mathcal{X}\to\mathcal{Z}$. Furthermore, G is a symmetry group acting on $\mathcal{X}$, $\cdot: G\times\mathcal{X}\to\mathcal{X}$, where the corresponding action on $\mathcal{Z}$ is $\cdot: G\times\mathcal{Z}\to\mathcal{Z}$ so that the symmetry in $\mathcal{X}$ is reflected in $\mathcal{Z}$. As a result a group action $g\in G$ commutes with the map $f(\mathbf{x})$

$$g\cdot f(\mathbf{x}) = f(g\cdot\mathbf{x}), \forall g\in G \text{ and } \mathbf{x}\in\mathcal{X}. \tag{1.14}$$

For example, a rotation in $\mathcal{X}$ has a corresponding transformation in $\mathcal{Z}$. Last, the representation $\mathbf{Z}$ is called disentangled with respect to the decomposition $G = G_1\times\cdots\times G_n$ if the following three conditions are satisfied

1. There is an action $g\cdot\mathbf{z}: G\times\mathcal{Z}\to\mathcal{Z}$.

2. The map $f:\mathcal{X}\to\mathcal{Z}$ is equivariant between the actions on $\mathcal{X}$ and $\mathcal{Z}$.

3. There is a decomposition $\mathbf{Z} = \mathbf{Z}_1\times\cdots\times\mathbf{Z}_n$ such that each $\mathbf{Z}_i$ is fixed by the action of all $G_{j,j\neq i}$ and affected only by G_i.

Note that the above definition of disentanglement also holds for semi-groups. Furthermore, we want to highlight the contrast to group-equivariant networks, see Section 1.2.1. Instead of choosing and enforcing a group symmetry *apriori*, disentanglement emerges during learning and depends heavily on the training data. In the best case, a diverse data set (different sizes, colors, orientations) of, e.g., natural objects [3] will lead to a learned representation where scale, color, and orientation are disentangled. For disentanglement, no expert knowledge is needed since the symmetries present in the data will be learned.

However, in contrast to group-equivariant networks, there is no guarantee for a disentangled representation to arise. At the time of writing, the exact cause of disentanglement in VAEs is poorly understood [Burgess et al., 2018]. Furthermore, Locatello et al. [2019] proof that the unsupervised learning of disentanglement is an ill-posed problem.

1.3.2 Normalizing flows

The second class of deep generative models that we consider are normalizing flows. Normalizing flows are based on the idea of transforming samples from a simple distribution into samples from a complex distribution using

[3] Natural objects are everyday objects like planes, cars, and cats.

the change of variables formula

$$p(\mathbf{x}) = p_{\mathbf{Z}}(f_{\boldsymbol{\theta}}(\mathbf{x})) \left| \det \left(\frac{\delta f_{\boldsymbol{\theta}}(\mathbf{x})}{\delta \mathbf{x}} \right) \right|, \tag{1.15}$$

where $\mathbf{z} = f_{\boldsymbol{\theta}}(\mathbf{x})$ is a bijective map $f : \mathcal{X} \to \mathcal{Z}$, $p_{\mathbf{Z}}(\mathbf{z})$ a simple prior distribution, and $\frac{\delta f_{\boldsymbol{\theta}}}{\delta \mathbf{x}}$ the Jacobian. With recent advances in architecture designs [Dinh et al., 2017], we are able to efficiently compute the determinant of the Jacobian for high dimensional $\mathbf{x}$. In contrast to VAEs which compute a variational approximation, see Section 1.3.1, normalizing flows let us estimate the marginal $p(\mathbf{x})$ exactly.

Furthermore, we are especially interested in conditional normalizing flows due to their relationship to SCMs. A slight modification of Equation 1.15 lets us learn conditional probabilities using normalizing flows. If we consider the conditioning variable Y Equation 1.15 becomes

$$p(\mathbf{x}|y) = p_{\mathbf{Z}}(f_{\boldsymbol{\theta},y}(\mathbf{x})) \left| \det \left(\frac{\delta f_{\boldsymbol{\theta},y}(\mathbf{x})}{\delta \mathbf{x}} \right) \right|, \tag{1.16}$$

where $f_{\boldsymbol{\theta},y}(\mathbf{x}) = f_{\boldsymbol{\theta}}(\mathbf{x}, y)$ is bijective in $\mathbf{X}$ but in general not in Y. Now, assume we want to learn the mapping between $\mathbf{X}$ and Y, assuming a non-linear causal relationship with graph $Y \to \mathbf{X}$. The corresponding SCM is given by

$$\mathbf{X} := f_{\mathbf{X}}(Y, \mathbf{N}_{\mathbf{X}}), \tag{1.17}$$

where $\mathbf{N}_{\mathbf{X}}$ is a noise variable independent of Y. Given samples $(\mathbf{x}, y)$ from the SCM above, we can learn to approximate the causal mechanism $f_{\mathbf{X}}$ using a normalizing flow. Since $f_{\boldsymbol{\theta},y}(\mathbf{x})$ is bijective in $\mathbf{x}$ by design, we can use its inverse to compute $\mathbf{x}$

$$\mathbf{x} = f^{-1}_{\boldsymbol{\theta},y}(\mathbf{z}), \mathbf{z} \sim p_{\mathbf{Z}}(\mathbf{z}), \tag{1.18}$$

where $p_{\mathbf{Z}}(\mathbf{z})$ is commonly chosen to be a standard Gaussian. The two functions $f_{\mathbf{X}}$ and $f_{\boldsymbol{\theta},y}$ are considered equivalent in this setting if for every value $Y = y$

$$f_{\mathbf{X}}(Y = y, \mathbf{N}_{\mathbf{X}}) \approx f^{-1}_{\boldsymbol{\theta}, Y=y}(\mathbf{Z}), \tag{1.19}$$

i.e., the two distributions are interventionally equivalent with respect to interventions on Y. This property makes normalizing flows a powerful tool to learn the invariant mechanisms of an SCM.

1.4 Research questions

We formulate the research questions and contributions in this work as follows.

Research Question 1: *How can we build a permutation-invariant deep learning architecture for set classification?*

A set is a collection of elements without any natural order, e.g., a collection of medical images. Therefore, any deep learning model for set classification needs to be invariant to changes in the order of the elements of a set. In general, a deep learning architecture for set classification consists of an encoder, pooling layer, and classifier. The encoder embeds each element of a set in a lower-dimensional space. Afterward, a pooling layer is used to combine the embeddings of all elements into one embedding capturing the characteristics of the whole set. Last, a classifier is used to map the set embedding to the label. Chapter 2 introduces a novel pooling layer based on the attention mechanisms. We show significant improvements in classification performance, especially in the low data regime, and improved interpretability, where the attention mechanism predicts a single weight for each element of the set. After training, these so-called attention weights can highlight the importance of each element of a set for the prediction task.

Research Question 2: *How does the approximate invariance enforced by data augmentation relate to invariant causal mechanisms?*

In Chapter 3, we show that data augmentation can be used to simulate interventions. As seen in Section 1.2.1, data augmentation is commonly interpreted as (semi)group transformations of the observational data, without any connection to causal mechanisms. Furthermore, in Section 1.2.2, we have shown that interventions need to be performed before the data generation. We show how models trained with data augmentation can learn invariant mechanisms by reinterpreting data augmentation as a tool to simulate interventions on purely observed data. We summarize our findings as a novel condition that we call *intervention-augmentation equivariance*. Last, based on those theoretical insights, we propose a simple algorithm automatically selecting data augmentation from a list of augmentation techniques. We test the proposed algorithm in a domain generalization setting and find strong generalization performance.

Research Question 3: *How to learn independent mechanisms from observational and interventional data jointly?*

As seen in Section 1.2.2, at least in theory, we can learn invariant mechanisms from unbiased data. However, the learned estimate might suffer from high variance due to the generally low sample size of interventional data sets. In Chapter 4, we introduce a novel approach of using additional observational data to reduce the variance without introducing confounding bias. Our contribution is twofold. First, we present a new theoretical tool for causal inference called *causal* reductions, where we can replace any number of confounders with a single latent confounder that lives in the same space as the direct cause. The resulting reduced causal model is

interventionally and observationally equivalent to the non-reduced one. Second, we propose an efficient parameterization of reduced causal models using normalizing flows. We exploit that some parameters are shared between observational and interventional distribution, which results in lower sample complexity.

Research Question 4: *How can we augment a variational autoencoder to learn domain invariant latent features reliably?*

In Chapter 5, we propose an augmented version of the variational autoencoder, see Section 1.3.1. To guide the disentanglement, we divide the latent space into three subspaces: one containing information shared among domains, one containing information about the differences among domains, and one for all remaining factors of variation. We call this model the Domain Invariant Variational Autoencoder (DIVA). Since the first latent subspace is invariant to domain changes, we can build a domain invariant classifier by ignoring the other two. Furthermore, we show that DIVA can be trained in a semi-supervised fashion due to its generative nature, where we do not have access to the labels of all samples of the training dataset.

2. Attention-based Deep Multiple Instance Learning

We start the central part of the thesis with what we consider the strongest type of invariance. As seen in Section 1.2.1, we can enforce invariance with respect to a symmetry group using the architecture of our deep learning model, where the choice of architecture will depend on the data and task. As a result, the model will be strictly invariant after training.

In the following, we will focus on the task of set classification. A set is a collection of unordered elements. Naturally, any machine learning model for set classification needs to be permutation invariant to the order of the set. We propose an attention-based pooling layer for set classification, where the attention mechanism is used to enforce invariance with respect to the permutation group. I.e., a change in the order of the input set leads to the same prediction as the original order.

2.1 Introduction

In typical machine learning problems like image classification, it is assumed that an image represents a category (a class). However, multiple instances are observed in many real-life applications, and only a general statement of the category is given. This scenario is called *multiple instance learning* (MIL) [Dietterich et al., 1997, Maron and Lozano-Pérez, 1998] or, *learning from weakly annotated data* [Oquab et al., 2014]. The problem of weakly annotated data is especially apparent in medical imaging [Quellec et al., 2017] (e.g., computational pathology, mammography or CT lung screening) where a single label typically describes an image (benign/malignant) or a Region Of Interest (ROI) is roughly given.

MIL deals with a bag of instances for which a single class label is assigned. Hence, the main goal of MIL is to learn a model that predicts a bag label, e.g., a medical diagnosis. An additional challenge is to discover *key instances* [Liu et al., 2012], i.e., the instances that trigger the bag label. In the medical domain, the latter task is of great interest because

of legal issues[1] and its usefulness in clinical practice. To solve the primary task of a bag classification, different methods are proposed, such as utilizing similarities among bags [Cheplygina et al., 2015b], embedding instances to a compact low-dimensional representation that is further fed to a bag-level classifier [Andrews et al., 2003, Chen et al., 2006], and combining responses of an instance-level classifier [Ramon and De Raedt, 2000, Raykar et al., 2008, Zhang et al., 2006]. Only the last approach is capable of providing interpretable results. However, it was shown that the instance level accuracy of such methods is low [Kandemir and Hamprecht, 2015] and in general, there is a disagreement among MIL methods at the instance level [Cheplygina et al., 2015a]. These issues call into question the usability of current MIL models for interpreting the final decision.

In this paper, we propose a new method that incorporates interpretability into the MIL approach and increases its flexibility. We formulate the MIL model using the Bernoulli distribution for the bag label and train it by optimizing the log-likelihood function. We show that the application of the Fundamental Theorem of Symmetric Functions provides a general procedure for modeling the bag label probability (the bag score function) that consists of three steps: (i) a transformation of instances to a low-dimensional embedding, (ii) a permutation-invariant (symmetric) aggregation function, and (iii) a final transformation to the bag probability. We propose to parameterize all transformations using neural networks (i.e., a combination of convolutional and fully-connected layers), which increases the flexibility of the approach and allows to train the model in an end-to-end manner by optimizing an unconstrained objective function. Last but not least, we propose to replace widely-used permutation-invariant operators such as the maximum operator max and the mean operator mean by a trainable weighted average. A two-layered neural network gives weights for each instance. The two-layered neural network corresponds to the attention mechanism [Bahdanau et al., 2014, Raffel and Ellis, 2016]. Notably, the attention weights allow us to find key instances, which could further highlight possible ROIs. The experiments show that our model is on par with the best classical MIL methods on standard benchmark MIL datasets. It outperforms other methods on an MNIST-based MIL problem and two real-life histopathology image datasets. Moreover, we provide empirical evidence that our model can indicate key instances.

[1] According to the European Union General Data Protection Regulation (taking effect 2018), a user should have the right to obtain an explanation of the decision reached.

2.2 Method

2.2.1 Multiple instance learning (MIL)

Problem formulation In the classical (binary) supervised learning problem one aims at finding a model that predicts a value of a target variable, $y \in \{0,1\}$, for a given instance, $\mathbf{x} \in \mathbb{R}^D$. In the case of the MIL problem, however, instead of a single instance there is a bag of instances, $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$, that exhibit neither dependency nor ordering among each other. We assume that K could vary for different bags. There is also a single binary label Y associated with the bag. Furthermore, we assume that individual labels exist for the instances within a bag, i.e., $y_1, \ldots, y_K$ and $y_k \in \{0,1\}$, for $k = 1, \ldots, K$, however, there is no access to those labels and they remain unknown during training. We can re-write the assumptions of the MIL problem in the following form:

$$Y = \begin{cases} 0, & \text{iff } \sum_k y_k = 0, \\ 1, & \text{otherwise.} \end{cases} \tag{2.1}$$

These assumptions imply that a MIL model must be **permutation - invariant**. Further, the two statements could be re-formulated in a compact form using the maximum operator:

$$Y = \max_k \{y_k\}. \tag{2.2}$$

Learning a model that tries to optimize an objective based on the maximum over instance labels would be problematic, at least for two reasons. First, all gradient-based learning methods would encounter issues with vanishing gradients. Second, this formulation is suitable only when an instance-level classifier is used.

To make the learning problem more straightforward, we propose to train a MIL model by optimizing the log-likelihood function where the bag label is distributed according to the Bernoulli distribution with the parameter $\theta(X) \in [0,1]$, i.e., the probability of $Y = 1$ given the bag of instances X.

MIL approaches In the MIL setting, the bag probability $\theta(X)$ must be permutation-invariant since we assume neither ordering nor dependency of instances within a bag. Therefore, the MIL problem can be considered in terms of a specific form of the Fundamental Theorem of Symmetric Functions with monomials given by the following theorem [Zaheer et al., 2017]:

Theorem 1. *A scoring function for a set of instances X, $S(X) \in \mathbb{R}$, is a symmetric function (i.e., permutation-invariant to the elements in X), if and only if it can be decomposed in the following form:*

$$S(X) = g\left(\sum_{\mathbf{x} \in X} f(\mathbf{x})\right), \tag{2.3}$$

where f and g are suitable transformations.

This theorem provides a general strategy for modeling the bag probability using the decomposition given in (2.3). A similar decomposition with max instead of sum is given by the following theorem [Qi et al., 2017]:

Theorem 2. *For any $\varepsilon > 0$, a Hausdorff continuous symmetric function $S(X) \in \mathbb{R}$ can be arbitrarily approximated by a function in the form $g(\max_{\mathbf{x}\in X} f(\mathbf{x}))$, where* max *is the element-wise vector maximum operator and f and g are continuous functions, that is:*

$$\left| S(X) - g\left(\max_{\mathbf{x}\in X} f(\mathbf{x})\right)\right| < \varepsilon. \tag{2.4}$$

The difference between Theorems 1 and 2 is that the former is a universal decomposition while the latter provides an arbitrary approximation. Nonetheless, they both formulate a general three-step approach for classifying a bag of instances: (i) a transformation of instances using the function f, (ii) a combination of transformed instances using a symmetric (permutation-invariant) function σ, (iii) a transformation of combined instances transformed by f using a function g. Finally, the expressiveness of the score function relies on the choice of classes of functions for f and g.

In the MIL problem formulation, the score function in both theorems is the probability $\theta(X)$, and the permutation-invariant function σ is referred to as the MIL pooling. The choice of functions f, g, and σ determines a specific approach to modeling the label probability. For a given MIL operator, there are two main MIL approaches:

(i) *The instance-level approach*: The transformation f is an instance-level classifier that returns scores for each instance. Then individual scores are aggregated by MIL pooling to obtain $\theta(X)$. The function g is the identity function.

(ii) *The embedding-level approach*: The function f maps instances to a low-dimensional embedding. MIL pooling is used to obtain a bag representation independent of the number of instances in the bag. A bag-level classifier further processes the bag representation to provide $\theta(X)$.

It is advocated in Wang et al. [2016] that the latter approach is preferable for the bag level classification performance. Since the individual labels are unknown, there is a threat that the instance-level classifier might be trained insufficiently, and it introduces additional errors to the final prediction. The embedding-level approach determines a joint representation of a bag, and therefore it does not introduce additional bias to the bag-level classifier. On the other hand, the instance-level approach provides a score that can be used to find *key instances* i.e., the instances that trigger the

bag label. Liu et al. [2012] were able to show that a model that is successfully detecting key instances is more likely to achieve better bag label predictions. We will show how to modify the embedding-level approach to be interpretable by using a new MIL pooling.

2.2.2 MIL with Neural Networks

In classical MIL problems it is assumed that instances are represented by features that do not require further processing, i.e., f is the identity. However, for some tasks like image or text analysis additional steps of feature extraction are necessary. Additionally, Theorem 1 and 2 indicate that for a flexible enough class of functions we can model any permutation-invariant score function. Therefore, we consider a class of transformations that are parameterized by neural networks $f_{\boldsymbol{\psi}}(\cdot)$ with parameters $\boldsymbol{\psi}$ that transform the k-th instance into a low-dimensional embedding, $\mathbf{h}_k = f_{\boldsymbol{\psi}}(\mathbf{x}_k)$, where $\mathbf{h}_k \in \mathcal{H}$ such that $\mathcal{H} = [0,1]$ for the instance-based approach and $\mathcal{H} = \mathbb{R}^M$ for the embedding-based approach.

Eventually, the parameter $\theta(X)$ is determined by a transformation $g_{\boldsymbol{\phi}} : \mathcal{H}^K \to [0,1]$. In the instance-based approach the transformation $g_{\boldsymbol{\phi}}$ is simply the identity, while in the embedding-based approach it could be also parameterized by a neural network with parameters $\boldsymbol{\phi}$. The former approach is depicted in Figure 2.6(a) and the latter in Figure 2.6(b) in the Appendix.

The idea of parameterizing all transformations using neural networks is very appealing because the whole approach can be arbitrarily flexible, and it can be trained end-to-end by backpropagation. The only restriction is that the MIL pooling must be differentiable.

2.2.3 MIL pooling

The formulation of the MIL problem requires the MIL pooling σ to be permutation-invariant. As shown in Theorem 1 and 2, there are two MIL pooling operators that ensure the score function (i.e., the bag probability) to be a symmetric function, namely, the maximum operator:

$$\forall_{m=1,\ldots,M} : z_m = \max_{k=1,\ldots,K} \{\mathbf{h}_{km}\}, \tag{2.5}$$

and the mean operator:[2]

$$\mathbf{z} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{h}_k. \tag{2.6}$$

In fact, other operators could be used such as, the convex maximum operator (i.e., log-sum-exp) [Ramon and De Raedt, 2000], Integrated Segmentation and Recognition [Keeler et al., 1991], noisy-or [Maron and

[2] Note that the weight $\frac{1}{K}$ can be seen as a part of the f function.

Lozano-Pérez, 1998] and noisy-and [Kraus et al., 2016]. These MIL pooling operators could replace max in Theorem 2 and proofs would follow similarly (see Supplementary in Qi et al. [2017] for a detailed proof for the maximum operator). These operators are differentiable; hence, they could be easily used as a MIL pooling layer in a deep neural network architecture.

2.2.4 Attention-based MIL pooling

All MIL pooling operators mentioned in the previous section have a clear disadvantage: they are pre-defined and non-trainable. For instance, the max-operator could be a good choice in the instance-based approach, but it might be inappropriate for the embedding-based approach. Similarly, the mean operator is a bad MIL pooling to aggregate instance scores, although it could succeed in calculating the bag representation. Therefore, a flexible and adaptive MIL pooling could potentially achieve better results by adjusting to a task and data. Ideally, such MIL pooling should also be interpretable, a trait that is missing in all operators mentioned in Section 2.2.3.

Attention mechanism We propose to use a weighted average of instances (low-dimensional embeddings) where weights are determined by a neural network. Additionally, the weights must sum to 1 to be invariant to the size of a bag. The weighted average fulfills the requirements of the Theorem 1 where the weights together with the embeddings are part of the f function. Let $H = \{\mathbf{h}_1, \dots, \mathbf{h}_K\}$ be a bag of K embeddings, then we propose the following MIL pooling:

$$\mathbf{z} = \sum_{k=1}^{K} a_k \mathbf{h}_k, \tag{2.7}$$

where:

$$a_k = \frac{\exp\left\{\mathbf{w}^\top \tanh\left(V\mathbf{h}_k^\top\right)\right\}}{\sum_{j=1}^{K} \exp\left\{\mathbf{w}^\top \tanh\left(V\mathbf{h}_j^\top\right)\right\}}, \tag{2.8}$$

where $\forall_{k=1,\dots,K}\ \mathbf{w}_k \in \mathbb{R}^{L\times 1}$ and $V \in \mathbb{R}^{L\times M}$ are parameters. Moreover, we utilize the hyperbolic tangent $\tanh(\cdot)$ element-wise non-linearity to include both negative and positive values for proper gradient flow. The proposed construction allows to discover (dis)similarities among instances.

Interestingly, the proposed MIL pooling corresponds to a version of the attention mechanism [Lin et al., 2017, Raffel and Ellis, 2016]. The main difference is that all instances are sequentially dependent for the attention mechanism, while we assume that all instances are independent. Therefore, a naturally arising question is whether the attention mechanism could work without sequential dependencies among instances and if it will not learn the mean operator. We will address this issue in the experiments.

Gated attention mechanism Furthermore, we notice that the $\tanh(\cdot)$ non-linearity could be inefficient to learn complex relations. Our concern

follows from the fact that $\tanh(x)$ is approximately linear for $x \in [-1, 1]$, which could limit the final expressiveness of learned relations among instances. Therefore, we propose to additionally use the gating mechanism [Dauphin et al., 2017] together with $\tanh(\cdot)$ non-linearity that yields:

$$a_k = \frac{\exp\left\{\mathbf{w}^\top \left(\tanh\left(V\mathbf{h}_k^\top\right) \odot \operatorname{sigm}\left(U\mathbf{h}_k^\top\right)\right)\right\}}{\sum_{j=1}^{K} \exp\left\{\mathbf{w}^\top \left(\tanh\left(V\mathbf{h}_j^\top\right) \odot \operatorname{sigm}\left(U\mathbf{h}_j^\top\right)\right)\right\}}, \tag{2.9}$$

where $U \in \mathbb{R}^{L\times M}$ are parameters, $\odot$ is an element-wise multiplication and $\operatorname{sigm}(\cdot)$ is the sigmoid non-linearity. The gating mechanism introduces a learnable non-linearity that potentially removes the troublesome linearity in $\tanh(\cdot)$.

Flexibility In principle, the proposed attention-based MIL pooling allows to assign different weights to instances within a bag. Hence, the final representation of the bag could be highly informative for the bag-level classifier. In other words, it should be able to find key instances. Moreover, applying the attention-based MIL pooling together with the transformations f and g parameterized by neural networks makes the whole model fully differentiable and adaptive. These two facts make the proposed MIL pooling a potentially very flexible operator that could model an arbitrary permutation-invariant score function. The proposed attention mechanism, together with a deep MIL model, is depicted in Figure 2.6(c) in the Appendix.

Interpretability Ideally, in the case of a positive label ($Y = 1$), high attention weights should be assigned to instances that are likely to have label $y_k = 1$ (key instances). Namely, the attention mechanism allows an interpretation of the provided decision in terms of instance-level labels easily. The attention network does not provide scores as the instance-based classifier does, but it can be considered a proxy. The attention-based MIL pooling bridges the instance-level approach and the embedding-level approach.

From the practical point of view, e.g., in the computational pathology, it is desirable to provide ROIs together with the final diagnosis to a doctor. Therefore, the attention mechanism is potentially of great interest in practical applications.

2.3 Related work

MIL pooling Typically, MIL approaches utilize either the mean pooling or the max pooling, while the latter is mostly used [Feng and Zhou, 2017, Pinheiro and Collobert, 2015, Zhu et al., 2017]. Both operators are non-trainable which potentially limits their applicability. Some other MIL pooling operators contain global adaptive parameters, such as noisy-and [Kraus et al., 2016]. However, their flexibility is restricted. We propose a

fully trainable MIL pooling that adapts to new instances.

MIL with neural networks In the classical work on MIL, it is assumed that pre-computed features represent instances, and there is very little need to apply additional feature extraction. Nevertheless, recent work on utilizing fully-connected neural networks in MIL shows that it could still be beneficial [Wang et al., 2016]. Similarly, in computer vision, the idea of MIL combined with deep learning significantly improves final accuracy [Oquab et al., 2014]. In this paper, we follow this line of research since it allows us to apply a flexible class of transformations that can be trained end-to-end by backpropagation.

MIL and attention The attention mechanism is widely used in deep learning for image captioning [Xu et al., 2015] or text analysis [Bahdanau et al., 2014, Lin et al., 2017]. For the MIL problem, it has rarely been used and only in a limited form. In Pappas and Popescu-Belis [2014] an attention-based MIL was proposed, but attention weights were trained as parameters of an auxiliary linear regression model. This idea was further expanded, and the linear regression model was replaced by a one-layer neural network with single output [Pappas and Popescu-Belis, 2017]. The attention-based MIL operator was used very recently in Qi et al. [2017]. However, the attention was calculated using the dot product, and it performed worse than the max operator. Here, we propose to use a two-layered neural network to learn the MIL operator, and we show that it outperforms commonly used MIL pooling operators.

MIL for medical imaging The MIL seems to perfectly fit medical imaging where processing a whole image consisting of billions of pixels is computationally infeasible. Moreover, in the medical domain, it is very laborious to obtain pixel-level annotations. Therefore, it is tempting to divide a medical image into smaller patches treated as a bag with a single label [Quellec et al., 2017]. This idea attracts great interest in computational histopathology, where patches could correspond to cells that indicate malignant changes [Sirinukunwattana et al., 2016]. Different MIL approaches were used for histopathology data, such as Gaussian processes [Kandemir et al., 2014, 2016] or a two-stage approach with neural networks and EM algorithm to determine instance classes [Hou et al., 2016]. Other applications of MIL methods in medical imaging are mammography (nodule) classification [Zhu et al., 2017] and microscopy cell detection [Kraus et al., 2016]. In this paper, we show that the proposed attention-based deep MIL approach can be used to provide the final diagnosis and indicate ROIs in a histopathology slide.

2.4 Experiments

In the experiments, we aim at evaluating the proposed approach: a MIL model parameterized with neural networks and a (gated) attention-based pooling layer ('Attention' and 'Gated-Attention'). We evaluate our approach on a number of different MIL datasets: five MIL benchmark datasets (MUSK1, MUSK2, FOX, TIGER, ELEPHANT), an MNIST-based image dataset (MNIST-BAGS) and two real-life histopathology datasets (BREAST CANCER, COLON CANCER). We want to verify two research questions in the experiments: (i) whether our approach achieves the best performance or is comparable to the best performing method, (ii) if our method can provide interpretable results by using the attention weights that indicate key instances or ROIs.

To obtain a fair comparison, we use a common evaluation methodology, i.e., 10-fold-cross-validation, and five repetitions per experiment. In the case of MNIST-BAGS, we use a fixed division into training and test set. To create test bags, we solely sampled images from the MNIST test set. During training, we only used images from the MNIST training set. For all experiments, we use modified versions of models that have shown high classification performance on the individual datasets [Wang et al., 2016, LeCun et al., 1998, Sirinukunwattana et al., 2016]. The MIL pooling layers are either located before the last layer of the model (the embedded-based approach) or after the last layer of the model (the instance-based approach). If an attention-based MIL pooling layer is used, the number of parameters in U was determined using a validation set. We tested the following dimensions (L): 64, 128, and 256. The different dimensions only resulted in minor changes in the model's performance. For layers using the gated attention mechanism, U and V have the same number of parameters. Finally, all layers were initialized according to Glorot and Bengio [2010] and biases were set to zero.

We compare our approach to various MIL methods on MIL benchmark datasets. On the image datasets, our method is compared with instance-level and embedding-level neural networks and commonly used MIL pooling layers (max and mean). In the following, we use 'Instance+max/mean' and 'Embedding+max/mean' to indicate networks built from convolutional and fully-connected layers. In contrast to networks purely build from fully-connected layers, referred to as 'mi-Net' and 'MI-Net' [Wang et al., 2016].

On MNIST-BAGS we include a SVM-based MIL model, called (MI-SVM). We do not present results of MI-SVM on the histopathology datasets since we could not train (including hyperparameter search and five times 10-fold-cross-validation procedure) the model in a reasonable amount of time.[3]

[3] Learning a single MI-SVM took approximately one week due to the large number of patches.

To compare the bag level performance, we use the following metrics: the classification accuracy, precision, recall, F-score, and the Area Under the receiver operating characteristic Curve (AUC).

2.4.1 Classical MIL datasets

Details In the first experiment, we aim at verifying whether our approach can compete with the best MIL methods on historically important benchmark datasets. Since all five datasets contain pre-computed features and only a small number of instances and bags, neural networks are most likely not well suited. First we predict drug activity (MUSK1 and MUSK2). A molecule has the desired drug effect if and only if one or more of its conformations bind to the target binding site. Since molecules can adopt multiple shapes, a bag is composed of shapes belonging to the same molecule [Dietterich et al., 1997]. The three remaining datasets, ELEPHANT, FOX and TIGER, contain features extracted from images. Each bag consists of a set of segments of an image. For each category, positive bags are images that contain the animal of interest, and negative bags are images that contain other animals [Andrews et al., 2003]. For detailed information on the number of bags, instances, and features in each dataset, see Section 2.6.3 in the Appendix.

In our experiments we use the same architecture, optimizer and hyperparameters as in the MI-Net model [Wang et al., 2016].

Table 2.1. Results on classical MIL datasets. Experiments were run 5 times and an average of the classification accuracy (± a standard error of a mean) is reported. [1] [Andrews et al., 2003], [2] [Gärtner et al., 2002], [3] [Zhang and Goldman, 2002] [4] [Zhou et al., 2009] [5] [Wei et al., 2017] [6] [Wang et al., 2016]

METHOD	MUSK1	MUSK2	FOX	TIGER	ELEPHANT
mi-SVM [1]	0.874±N/A	0.836±N/A	0.582±N/A	0.784±N/A	0.822±N/A
MI-SVM [1]	0.779±N/A	0.843±N/A	0.578±N/A	0.840±N/A	0.843±N/A
MI-Kernel [2]	**0.880**±0.031	**0.893**±0.015	**0.603**±0.028	0.842±0.010	0.843±0.016
EM-DD [3]	0.849±0.044	**0.869**±0.048	**0.609**±0.045	0.730±0.043	0.771±0.043
mi-Graph [4]	**0.889**±0.033	**0.903**±0.039	**0.620**±0.044	**0.860**±0.037	**0.869**±0.035
miVLAD [5]	**0.871**±0.043	**0.872**±0.042	**0.620**±0.044	0.811±0.039	**0.850**±0.036
miFV [5]	**0.909**±0.040	**0.884**±0.042	**0.621**±0.049	0.813±0.037	**0.852**±0.036
mi-Net [6]	**0.889**±0.039	**0.858**±0.049	**0.613**±0.035	0.824±0.034	**0.858**±0.037
MI-Net [6]	**0.887**±0.041	**0.859**±0.046	**0.622**±0.038	**0.830**±0.032	**0.862**±0.034
MI-Net with DS [6]	**0.894**±0.042	**0.874**±0.043	**0.630**±0.037	**0.845**±0.039	**0.872**±0.032
MI-Net with RC [6]	**0.898**±0.043	**0.873**±0.044	**0.619**±0.047	**0.836**±0.037	**0.857**±0.040
Attention	**0.892**±0.040	**0.858**±0.048	**0.615**±0.043	**0.839**±0.022	**0.868**±0.022
Gated-Attention	**0.900**±0.050	**0.863**±0.042	**0.603**±0.029	**0.845**±0.018	**0.857**±0.027

Results and discussion The results of the experiment are presented in Table 2.1. Our approaches (Attention and Gated-Attention) are comparable

with the best performing classical MIL methods (notice the standard error of the mean).

2.4.2 MNIST-bags

Details The main disadvantage of the classical MIL benchmark datasets is that pre-computed features represent instances. To consider a more challenging scenario, we propose investigating a dataset created using the well-known MNIST image dataset. A bag comprises a random number of 28×28 grayscale images taken from the MNIST dataset. The number of images in a bag is Gaussian-distributed, and the closest integer value is taken. A bag is given a positive label if it contains one or more images labeled '9'. We chose '9' since it can be easily mistaken with '7' or '4'. We investigate the influence of the number of bags in the training set and the average number of instances per bag on the prediction performance. During the evaluation, we use a fixed number of 1000 test bags. For all experiments a LeNet5 model is used [LeCun et al., 1998], see Table 2.8 and 2.9 in the Appendix. The models are trained with the Adam optimization algorithm [Kingma and Ba, 2015]. We keep the default parameters for β_1 and β_2, see Table 2.10 in the Appendix. In addition, we compare our method with a SVM-based MIL method (MI-SVM) [Andrews et al., 2003] that uses a Gaussian kernel on raw pixel features[4].

In the experiments, we use different numbers of the mean bag size, namely, 10, 50, and 100, and the variance $2, 10, 20$, respectively. Moreover, we use varying numbers of training bags, i.e., $50, 100, 150, 200, 300, 400, 500$. These different settings allow us to verify how a different number of training bags and a different number of instances influence MIL models. We compare instance-based and embedding-based approaches parameterized with a neural network (LeNet5) with mean and max MIL pooling. We use AUC as the evaluation metric.

Results and discussion The results of AUC for the mean bag sizes equal to 10, 50 and 100 are presented in Figure 2.1, 2.2 and 2.3, respectively, and detailed results are given in the Appendix. The findings of the experiment are the following: First, the proposed attention-based deep MIL approach performs much better than other methods in the small sample size regime. Moreover, when there is a small effective size of the training set that corresponds to 50-150 bags for around 10 instances per bag (see Figure 2.1) or 50-100 bags in the case of on average 50 instances in a bag (see Figure 2.2), our method still achieves significantly higher AUC than all other methods. Second, we notice that our approach is more flexible and obtained better results than the SVM-based approach in all cases except large effective sample sizes (see Figure 2.3). Third, the embedding-based

[4] We use code provided with [Doran and Ray, 2014]: `https://github.com/garydoranjr/misvm`

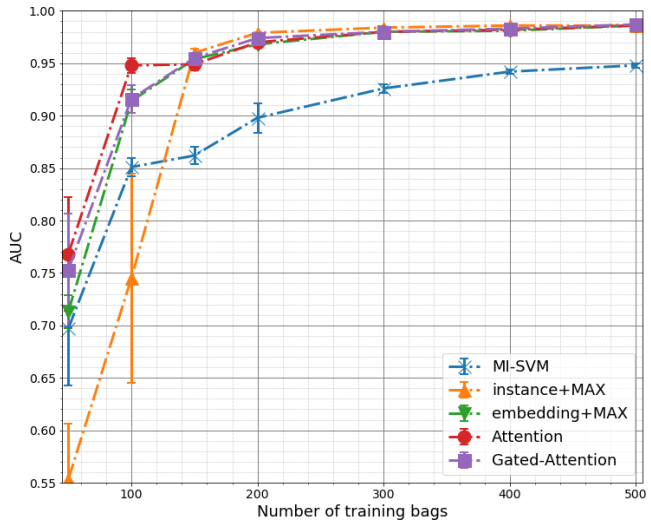


Figure 2.1. The test AUC for MNIST-BAGS with on average 10 instances per bag.

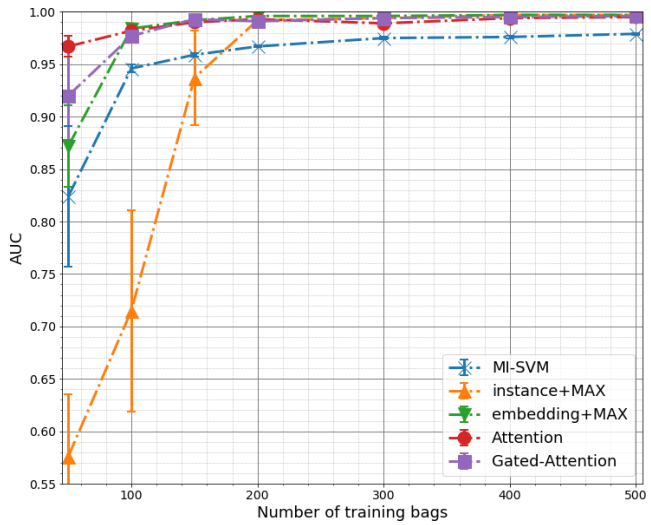


Figure 2.2. The test AUC for MNIST-BAGS with on average 50 instances per bag.

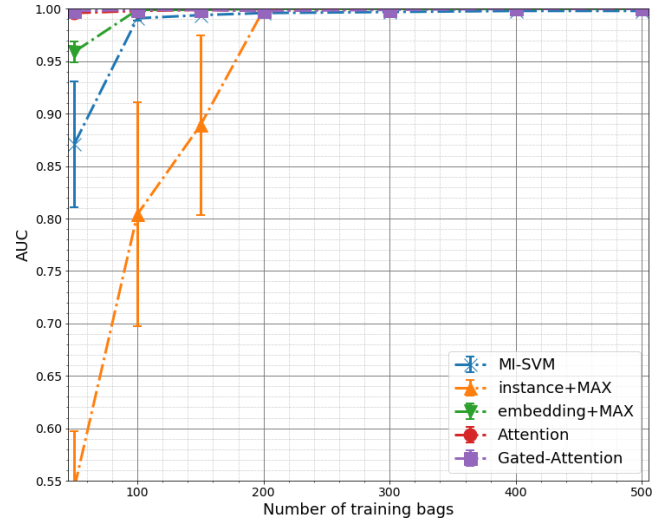


Figure 2.3. The test AUC for MNIST-BAGS with on average 100 instances per bag.

models performed better than the instance-based models. However, for a sufficient number of training images (number of training bags and training instances per bag) all models achieve very similar results. Fourth, the mean operator performs significantly worse than the max operator. However, the embedding-based model with the mean operator converged eventually to the best value but always later than the one with max. See Section 2.6.4 in the Appendix for details.

The results of this experiment indicate that for a small sample size regime, our approach is preferable to others. Since attention serves as a gradient update filter during backpropagation [Wang et al., 2017], instances with higher weights will contribute more to learning the encoder network of instances. This is especially important since medical imaging problems contain only a small number of cases. The more instances are in a bag, the easier the MIL task becomes since the MIL assumption states that every instance in a negative bag is negative. For example, a negative bag of size 100 from the MNIST-bags dataset will include about 11 negative examples per class.

Finally, we present an exemplary result of the attention mechanism in Figure 2.4. In this example, a bag consists of 13 images. For each digit, the corresponding attention weight is given by the trained network. The bag is correctly predicted as positive, and all nines are correctly highlighted. Hence, the attention mechanism works as expected. More examples are given in the Appendix.

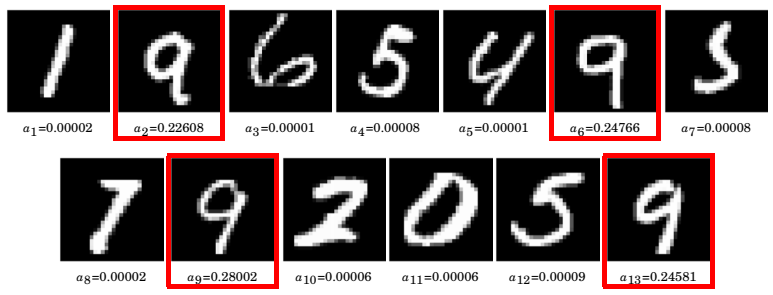


Figure 2.4. Example of attention weights for a positive bag.

2.4.3 Histopathology datasets

Details An automatic detection of cancerous regions in Hematoxylin and Eosin (H&E) stained whole-slide images is a task with high clinical relevance. Current supervised approaches utilize pixel-level annotations [Litjens et al., 2017]. However, data preparation requires a significant amount of time from pathologists, interfering with their daily routines. Hence, a successful solution working with weak labels would hold a great promise to reduce the workload of the pathologists. In the following, we perform two experiments on classifying weakly-labeled real-life histopathology images of the breast cancer dataset (BREAST CANCER) [Gelasca et al., 2008] and the colon cancer dataset (COLON CANCER) [Sirinukunwattana et al., 2016].

BREAST CANCER consists of 58 weakly labeled 896×768 H&E images. An image is labeled malignant if it contains breast cancer cells. Otherwise, it is benign. We divide every image into 32×32 patches. This results in 672 patches per bag. A patch is discarded if it contains 75% or more of white pixels.

COLON CANCER comprises 100 H&E images. The images originate from a variety of tissue appearances from both normal and malignant regions. For every image, the majority of nuclei of each cell were marked. In total, there are 22,444 nuclei with an associated class label, i.e. epithelial, inflammatory, fibroblast, and miscellaneous. A bag is composed of 27×27 patches. Furthermore, a bag is given a positive label if it contains one or more nuclei from the epithelial class. Tagging epithelial cells is highly relevant from a clinical point of view since colon cancer originates from epithelial cells [Ricci-Vitiani et al., 2007].

For both datasets we use the model proposed in Sirinukunwattana et al. [2016] for the transformation f. All models are trained with the Adam optimization algorithm [Kingma and Ba, 2015]. Due to the limited amount of data samples in both datasets we performed data augmentation to prevent overfitting. See the Appendix for further details.

Table 2.2. Results on BREAST CANCER. Experiments were run 5 times and an average (± a standard error of the mean) is reported.

METHOD	ACCURACY	PRECISION	RECALL	F-SCORE	AUC
Instance+max	0.614±0.020	0.585±0.03	0.477±0.087	0.506±0.054	0.612±0.026
Instance+mean	0.672±0.026	0.672±0.034	0.515±0.056	0.577±0.049	0.719±0.019
Embedding+max	0.607±0.015	0.558±0.013	0.546±0.070	0.543±0.042	0.650±0.013
Embedding+mean	**0.741**±0.023	**0.741**±0.023	0.654±0.054	0.689±0.034	**0.796**±0.012
Attention	**0.745**±0.018	0.718±0.021	**0.715**±0.046	**0.712**±0.025	0.775±0.016
Gated-Attention	**0.755**±0.016	**0.728**±0.016	**0.731**±0.042	**0.725**±0.023	**0.799**±0.020

Table 2.3. Results on COLON CANCER. Experiments were run 5 times and an average (± a standard error of the mean) is reported.

METHOD	ACCURACY	PRECISION	RECALL	F-SCORE	AUC
Instance+max	0.842 ± 0.021	0.866 ± 0.017	0.816 ± 0.031	0.839 ± 0.023	0.914 ± 0.010
Instance+mean	0.772 ± 0.012	0.821 ± 0.011	0.710 ± 0.031	0.759 ± 0.017	0.866 ± 0.008
Embedding+max	0.824 ± 0.015	0.884 ± 0.014	0.753 ± 0.020	0.813 ± 0.017	0.918 ± 0.010
Embedding+mean	0.860 ± 0.014	0.911 ± 0.011	0.804 ± 0.027	0.853 ± 0.016	0.940 ± 0.010
Attention	**0.904** ± 0.011	**0.953** ± 0.014	**0.855** ± 0.017	**0.901** ± 0.011	**0.968** ± 0.009
Gated-Attention	**0.898** ± 0.020	**0.944** ± 0.016	**0.851** ± 0.035	**0.893** ± 0.022	**0.968** ± 0.010

Results and discussion We present results in Table 2.2 and 2.3 for BREAST CANCER and COLON CANCER, respectively. First, we notice that the obtained results confirm our findings in MNIST-BAGS experiment that our approach outperforms all other methods. This trend is especially visible in the small-sample size regime of the MNIST-BAGS. Surprisingly, the embedding-based method with the max pooling failed almost completely on BREAST CANCER. Still, this dataset is generally difficult due to the high variability of slides and the small number of cases. The proposed method is not only the most accurate, but it also received the highest recall. A high recall is especially important in the medical domain since false negatives could lead to severe consequences, including patient fatality. We also notice that the gated-attention mechanism performs better than the basic attention mechanism on BREAST CANCER while these two behave similarly on COLON CANCER.

Eventually, we present the usefulness of the attention mechanism in providing ROIs. In Figure 2.5 we show a histopathology image divided into patches containing (mostly) single cells. We create a heatmap by multiplying patches by their corresponding attention weight. Although only image-level annotations are used during training, there is a substantial matching between the heatmap in Figure 2.5(d) and the ground truth in Figure 2.5(c). Additionally, we notice that the instance-based classifier tends to select only a small subset of positive patches (see Figure 2.10(e) in Appendix) that confirms low instance accuracy of the instance-based ap-

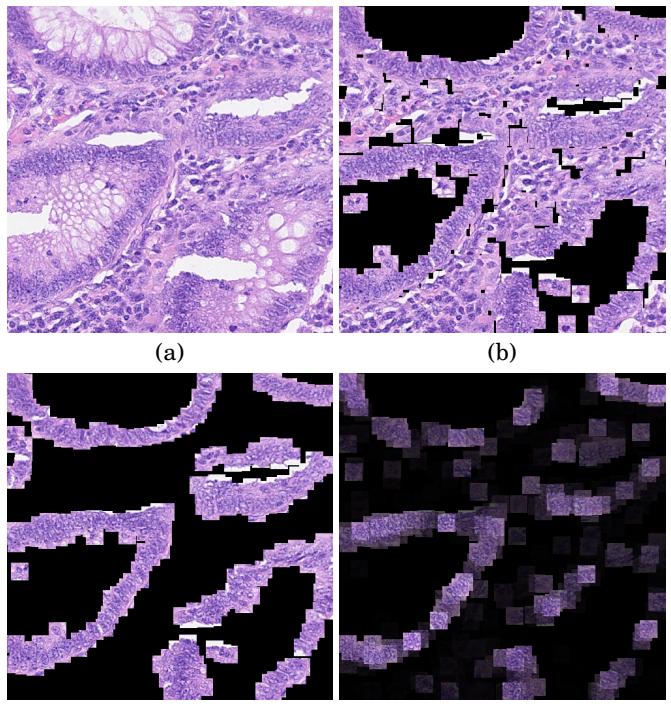


Figure 2.5. Visualization of the attention weights. (a) H&E stained histopathology image. (b) 27×27 patches centered around all marked nuclei. (c) Ground truth: Patches that belong to the class epithelial. (d) Heatmap: Every patch from (b) multiplied by its corresponding attention weight, we rescaled the attention weights using $a'_k = a_k - \min(\mathbf{a})/(\max(\mathbf{a}) - \min(\mathbf{a}))$.

proach discussed in Kandemir and Hamprecht [2015]. For more examples, please see the Appendix.

The obtained results again confirm that the proposed approach attains high predictive performance and highlights ROIs correctly. Moreover, the attention weights can be used to create a reliable heatmap.

2.5 Conclusion

In this paper, we proposed a flexible and interpretable MIL approach fully parameterized by neural networks. We outlined the usefulness of deep learning for modeling a permutation-invariant bag score function in terms of the Fundamental Theorem of Symmetric Functions. Moreover, we presented a trainable MIL pooling based on the (gated) attention mechanism. We showed empirically on five MIL datasets, one image corpora, and two real-life histopathology datasets that our method is on a par with the best

performing methods or performs the best in terms of different evaluation metrics. Additionally, we showed that our approach provides an interpretation of the decision by presenting ROIs, which is extremely important in many practical applications.

We strongly believe that the presented line of research is worth pursuing further. Here we focused on a binary MIL problem. However, the multi-class MIL is more interesting and challenging [Feng and Zhou, 2017]. Moreover, in some applications it is worth to consider *repulsion points* [Scott et al., 2005], i.e., instances for which a bag is always negative, or assume dependencies among instances within a bag [Zhou et al., 2009]. We leave investigating these issues for future research.

2.6 Appendix

2.6.1 Deep MIL approaches

In Figure 2.6 we present three deep MIL approaches discussed in the paper.

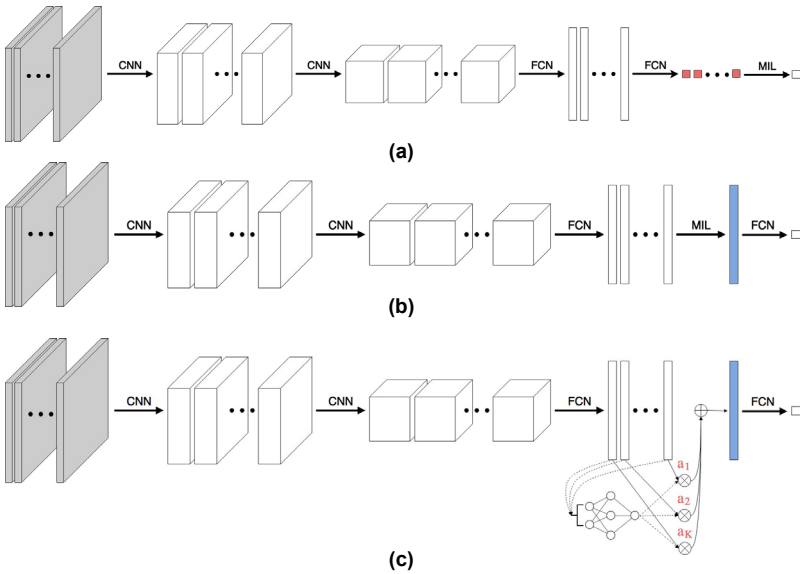


Figure 2.6. Deep MIL approaches. (a) the instance-based approach, (b) the embedding-based approach, (c) the proposed approach with the attention mechanism as the MIL pooling. Red color corresponds to instance scores, blue color depicts a bag vector representation. *Best viewed in color.*

2.6.2 Code

The implementation of our methods is available online at `https://github.com/AMLab-Amsterdam/AttentionDeepMIL`. All experiments were run on NVIDIA TITAN X Pascal with a batch size of 1 (= 1 bag) for all datasets.

2.6.3 Classical MIL datasets

Additional details In Table 2.1 a general description of the five benchmark MIL datasets used in the experiments is given. In Tables 2.5 and 2.6 we present architectures of the embedding-based and the instance-based models, respectively. We denote a fully-connected layer by 'fc', and the number of output hidden units is provided after a dash. The ReLU non-linearity was used. In Table 2.7 the details of the optimization (learning) procedure are given. We provide values of hyperparameters determined by the model selection procedure for which the highest validation performance was achieved.

Table 2.4. Overview of classical MIL datasets.

Dataset	# of bags	# of instances	# of features
Musk1	92	476	166
Musk2	102	6598	166
Tiger	200	1220	230
Fox	200	1302	230
Elephant	200	1391	230

Table 2.5. Classical MIL datasets: The embedding-based model architecture [Wang et al., 2016].

Layer	Type
1	fc-256 + ReLU
2	dropout
3	fc-128 + ReLU
4	dropout
5	fc-64 + ReLU
6	dropout
7	mil-max/mil-mean/mil-attention-64
8	fc-1 + sigm

Table 2.6. Classical MIL datasets: The instance-based model architecture [Wang et al., 2016].

Layer	Type
1	fc-256 + ReLU
2	dropout
3	fc-128 + ReLU
4	dropout
5	fc-64 + ReLU
6	dropout
7	fc-1 + sigm
8	mil-max/mil-mean

2.6.4 MNIST-bags

Additional details In Tables 2.8 and 2.9 we present architectures of the embedding-based and the instance-based models for MNIST-BAGS, respectively. We denote a convolutional layer by 'conv'. In brackets, we provide

Table 2.7. Classical MIL datasets: The optimization procedure details [Wang et al., 2016].

Experiment	Optimizer	Momentum	Learning rate	Weight decay	Epochs	Stopping criteria
Musk1	SGD	0.9	0.0005	0.005	100	lowest validation error and loss
Musk2	SGD	0.9	0.0005	0.03	100	lowest validation error and loss
Tiger	SGD	0.9	0.0001	0.01	100	lowest validation error and loss
Fox	SGD	0.9	0.0005	0.005	100	lowest validation error and loss
Elephant	SGD	0.9	0.0001	0.005	100	lowest validation error and loss

kernel size, stride, and padding. Kernels are provided after a dash. The convolutional max-pooling layer is denoted by 'maxpool', and the pooling size is given in brackets. The ReLU non-linearity was used. In Table 2.10 the details of the optimization (learning) procedure for deep MIL approach are given. The details of the SVM are given in Table 2.11. We provide values of hyperparameters determined by the model selection procedure for which the highest validation performance was achieved.

Table 2.8. MNIST-bags: The embedding-based model architecture [LeCun et al., 1998].

Layer	Type
1	conv(5,1,0)-20 + ReLU
2	maxpool(2,2)
3	conv(5,1,0)-50 + ReLU
4	maxpool(2,2)
5	fc-500 + ReLU
6	mil-max/mil-mean/mil-attention-128
7	fc-1 + sigm

Table 2.9. MNIST-bags: The instance-based model architecture [LeCun et al., 1998].

Layer	Type
1	conv(5,1,0)-20 + ReLU
2	maxpool(2,2)
3	conv(5,1,0)-50 + ReLU
4	maxpool(2,2)
5	fc-500 + ReLU
6	fc-1 + sigm
7	mil-max/mil-mean

Table 2.10. MNIST-bags: The optimization procedure details.

Experiment	Optimizer	β_1, β_2	Learning rate	Weight decay	Epochs	Stopping criteria
All	Adam	0.9, 0.999	0.0005	0.0001	200	lowest validation error+loss

Additional results In Tables 2.12, 2.13 and 2.14 we present the test AUC value for 10, 50 and 100 instances on average per a bag, respectively.

Table 2.11. MNIST-bags: SVM configuration.

Model	Features	Kernel	C	γ	Max iterations
MI-SVM	Raw pixel values	RBF	5	0.0005	200

In Figure 2.7 a negative bag is presented. In Figure 2.8 a positive bag with a single '9' is given. In Figure 2.9 a positive bag with multiple '9's is presented. In all figures attention weights are provided and in the case of positive bags a red rectangle highlights positive instances.

Table 2.12. The test AUC for MNIST-BAGS with on average 10 instances per bag for different numbers of training bags.

# train bags	50	100	150	200	300	400	500
Inst+max	0.553 ± 0.053	0.745 ± 0.100	0.960 ± 0.004	0.979 ± 0.001	0.984 ± 0.001	0.986 ± 0.001	0.986 ± 0.001
Inst+mean	0.663 ± 0.014	0.676 ± 0.012	0.694 ± 0.010	0.694 ± 0.017	0.709 ± 0.020	0.693 ± 0.023	0.712 ± 0.018
MI-SVM	0.697 ± 0.054	0.851 ± 0.009	0.862 ± 0.008	0.898 ± 0.014	0.926 ± 0.004	0.942 ± 0.002	0.948 ± 0.002
Embed+max	0.713 ± 0.016	0.914 ± 0.011	0.954 ± 0.005	0.968 ± 0.001	0.980 ± 0.001	0.981 ± 0.003	0.986 ± 0.002
Embed+mean	0.695 ± 0.026	0.841 ± 0.027	0.926 ± 0.004	0.953 ± 0.004	0.974 ± 0.002	0.980 ± 0.001	0.984 ± 0.002
Attention	0.768 ± 0.054	0.948 ± 0.007	0.949 ± 0.006	0.970 ± 0.003	0.980 ± 0.000	0.982 ± 0.001	0.986 ± 0.001
Gat Attention	0.753 ± 0.054	0.916 ± 0.013	0.955 ± 0.003	0.974 ± 0.002	0.980 ± 0.004	0.983 ± 0.002	0.987 ± 0.001

Table 2.13. The test AUC for MNIST-BAGS with on average 50 instances per bag for different numbers of training bags.

# train bags	50	100	150	200	300	400	500
Inst+max	0.576 ± 0.059	0.715 ± 0.096	0.937 ± 0.045	0.992 ± 0.002	0.994 ± 0.001	0.997 ± 0.001	0.997 ± 0.001
Inst+mean	0.737 ± 0.014	0.744 ± 0.029	0.824 ± 0.012	0.813 ± 0.030	0.722 ± 0.021	0.728 ± 0.017	0.798 ± 0.011
MI-SVM	0.824 ± 0.067	0.946 ± 0.004	0.959 ± 0.002	0.967 ± 0.002	0.975 ± 0.001	0.976 ± 0.001	0.979 ± 0.001
Embed+max	0.872 ± 0.039	0.984 ± 0.005	0.992 ± 0.001	0.996 ± 0.001	0.996 ± 0.001	0.997 ± 0.001	0.997 ± 0.001
Embed+mean	0.841 ± 0.013	0.906 ± 0.046	0.983 ± 0.005	0.992 ± 0.001	0.996 ± 0.001	0.997 ± 0.001	0.997 ± 0.001
Attention	0.967 ± 0.010	0.982 ± 0.003	0.990 ± 0.002	0.993 ± 0.002	0.989 ± 0.003	0.994 ± 0.001	0.995 ± 0.001
Gat Attention	0.920 ± 0.042	0.977 ± 0.006	0.993 ± 0.003	0.991 ± 0.002	0.994 ± 0.002	0.995 ± 0.001	0.996 ± 0.001

Table 2.14. The test AUC for MNIST-BAGS with on average 100 instances per bag for different numbers of training bags.

# train bags	50	100	150	200	300	400	500
Inst+max	0.543 ± 0.054	0.804 ± 0.107	0.899 ± 0.086	0.999 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Inst+mean	0.842 ± 0.023	0.855 ± 0.025	0.824 ± 0.014	0.896 ± 0.037	0.859 ± 0.029	0.899 ± 0.012	0.868 ± 0.016
MI-SVM	0.871 ± 0.060	0.991 ± 0.002	0.994 ± 0.002	0.996 ± 0.001	0.997 ± 0.001	0.998 ± 0.001	0.998 ± 0.001
Embed+max	0.977 ± 0.009	0.999 ± 0.001	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Embed+mean	0.959 ± 0.010	0.990 ± 0.003	0.998 ± 0.001	0.900 ± 0.089	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Attention	0.996 ± 0.001	0.998 ± 0.001	0.999 ± 0.000	0.998 ± 0.001	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Gat Attention	0.998 ± 0.001	0.999 ± 0.000	0.998 ± 0.001	0.998 ± 0.001	0.999 ± 0.000	1.000 ± 0.000	1.000 ± 0.000

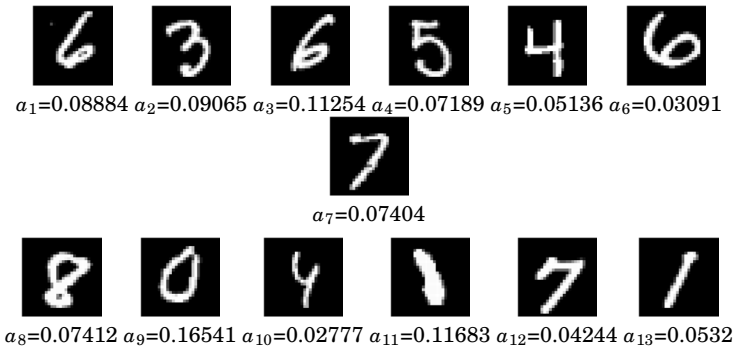


Figure 2.7. Example of attention weights for a negative bag.

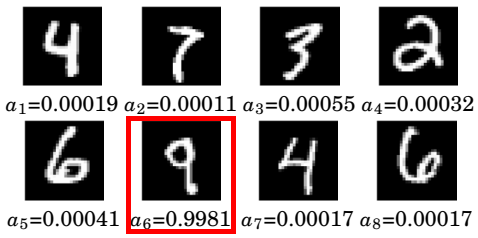


Figure 2.8. Example of attention weights for a positive bag containing a single '9'.

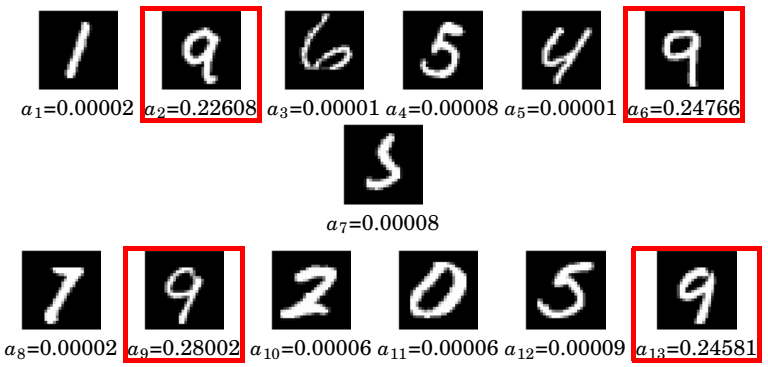


Figure 2.9. Example of attention weights for a positive bag containing multiple '9's.

2.6.5 Histopathology datasets

Data augmentation We randomly adjust the amount of H&E by decomposing the RGB color of the tissue into the H&E color space [Ruifrok and Johnston, 2001], followed by multiplying the magnitude of H&E for a pixel by two i.i.d. Gaussian random variables with expectation equal to one. We randomly rotate and mirror every patch. Lastly, we perform color normalization on every patch.

Additional details In Tables 2.15 and 2.16 we present architectures of the embedding-based and the instance-based models for histopathology datasets, respectively. In Table 2.17 the details of the optimization (learning) procedure for deep MIL approach are given. We provide values of hyperparameters determined by the model selection procedure for which the highest validation performance was achieved.

Table 2.15. Histopathology: The embedding-based model architecture [Sirinukunwattana et al., 2016].

Layer	Type
1	conv(4,1,0)-36 + ReLU
2	maxpool(2,2)
3	conv(3,1,0)-48 + ReLU
4	maxpool(2,2)
5	fc-512 + ReLU
6	dropout
7	fc-512 + ReLU
8	dropout
9	mil-max/mil-mean/mil-attention-128
10	fc-1 + sigm

Table 2.16. Histopathology: The instance-based model architecture [Sirinukunwattana et al., 2016].

Layer	Type
1	conv(4,1,0)-36 + ReLU
2	maxpool(2,2)
3	conv(3,1,0)-48 + ReLU
4	maxpool(2,2)
5	fc-512 + ReLU
6	dropout
7	fc-512 + ReLU
8	dropout
9	fc-1 + sigm
10	mil-max/mil-mean

Table 2.17. Histopathology: The optimization procedure details.

Experiment	Optimizer	β_1, β_2	Learning rate	Weight decay	Epochs	Stopping criteria
All	Adam	0.9, 0.999	0.0001	0.0005	100	lowest validation error+loss

Additional results In Figures 2.10, 2.11 and 2.12 five images are presented: (a) a full H&E image, (b) all patches containing cells, (c) positive patches, (d) a heatmap given by the attention mechanism, (e) a heatmap given by the Instance+max.

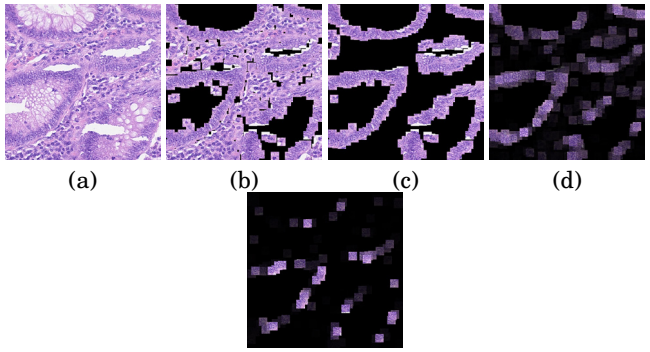


Figure 2.10. Colon cancer example 1. (a) H&E stained histopathology image. (b) 27×27 patches centered around all marked nuclei. (c) Ground truth: Patches that belong to the class epithelial. (d) Attention heatmap: Every patch from (b) multiplied by its attention weight. (e) Instance+max heatmap: Every patch from (b) multiplied by its score from the Instance+max model. We rescaled the attention weights and instance scores using $a'_k = a_k - \min(\mathbf{a})/(\max(\mathbf{a}) - \min(\mathbf{a}))$.

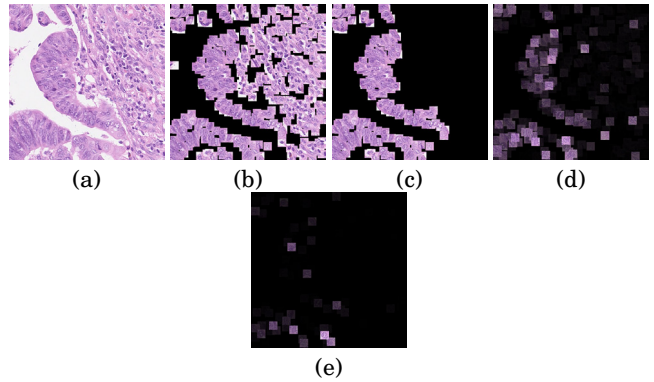


Figure 2.11. Colon cancer example 2. (a) H&E stained histopathology image. (b) 27×27 patches centered around all marked nuclei. (c) Ground truth: Patches that belong to the class epithelial. (d) Attention heatmap: Every patch from (b) multiplied by its attention weight. (e) Instance+max heatmap: Every patch from (b) multiplied by its score from the Instance+max model. We rescaled the attention weights and instance scores using $a'_k = a_k - \min(\mathbf{a})/(\max(\mathbf{a}) - \min(\mathbf{a}))$.

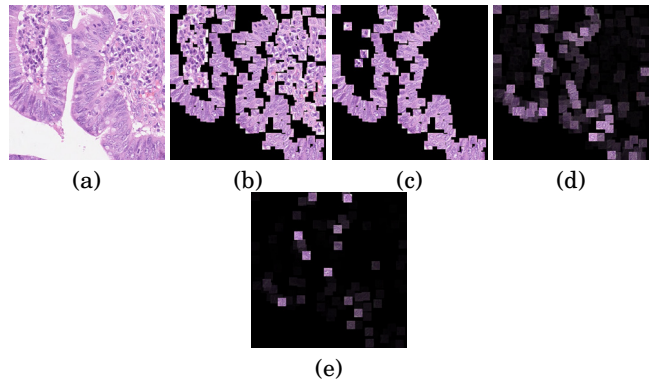


Figure 2.12. Colon cancer example 3. (a) H&E stained histopathology image. (b) 27×27 patches centered around all marked nuclei. (c) Ground truth: Patches that belong to the class epithelial. (d) Attention heatmap: Every patch from (b) multiplied by its attention weight. (e) Instance+max heatmap: Every patch from (b) multiplied by its score from the Instance+max model. We rescaled the attention weights and instance scores using $a'_k = a_k - \min(\mathbf{a})/(\max(\mathbf{a}) - \min(\mathbf{a}))$.

3. Selecting Data Augmentation for Simulating Interventions

In the previous section, we have shown how to enforce invariance with respect to a specific symmetry group by altering the architecture of a deep learning model. However, for many practical applications, one of the following two problems arises: (i) the required symmetry is not a group, but a semigroup, (ii) no architecture exists that can enforce the required symmetry.

In applications like this, data augmentation can often be used to enforce approximate invariance. In contrast to group-equivariant neural networks, see Section 1.2.1, data augmentation can be used to enforce invariance with respect to semigroups like color transformations and occlusion. Since data augmentation is a transformation of the input, it is independent of the architecture of the deep learning model.

The present section will show that data augmentation is a powerful tool to train machine learning models that generalize across domains. By adopting a causal point of view, we can relate data augmentation to interventions. As a result, we can explain the success of data augmentation from a causal perspective by connecting the two notions of invariance introduced in Section 1.

3.1 Introduction

Despite recent advancements in machine learning fueled by deep learning, studies like Azulay and Weiss [2019] have shown that deep learning methods may not generalize to inputs from outside of their training distribution. In safety-critical fields like medical imaging, robotics, and self-driving cars, machine learning models must be robust to changes in the environment. Without the ability to generalize, machine learning models cannot be safely deployed in the real world.

In the field of domain generalization, one tries to find a representation that generalizes across different environments, called *domains*, each with a distinct shift of the input. This problem is especially challenging when

changes in the domain are spuriously correlated with changes in the actual task labels, e.g., due to a biased data gathering process. An example is given by Arjovsky et al. [2020]: If we consider a dataset of images of cows and camels in their natural habitat, then there is a strong correlation between the type of animal and the landscape in the image, e.g., a camel standing in a desert. If we now train a machine learning model to predict the animal in a given image, the model is prone to exploit the spurious correlation between the type of animal and the type of landscape. As a result, the model can fail to recognize a camel standing in a green pasture or a cow standing in a desert.

A large corpus of methods designed to learn representations that will generalize across domains has been formulated in recent years. While the proposed methods can achieve good results on various domain generalization benchmarks, most lack a theoretical foundation. In the worst-case scenario, these methods enforce the wrong type of invariance, as proven in Appendix 3.6.8 . Researchers have found a practical way of dealing with the spurious correlation between domains and the actual task, especially in more applied fields, like medical imaging and robotics. Data augmentation in combination with Empirical Risk Minimization (ERM) [Vapnik, 1992] is used to enforce invariance of the machine learning model to changes in the domain. The appropriate data augmentation is selected using prior knowledge. In Appendix 3.6.10, we give a detailed summary of two successful applications of data augmentation in the context of domain generalization.

However, the success of data augmentation is often described in vague terms like 'artificially expanding labeled training datasets' [Li, 2020] and 'reduce overfitting' [Krizhevsky et al., 2012]. In this paper, we present a causal perspective on data augmentation in the context of domain generalization and contribute to the field in the following manner:

- First, we introduce the concept of *intervention-augmentation equivariance* that formalizes the relationship between data augmentation and interventions on features caused by the domain. We show that if intervention-augmentation equivariance holds, we can use data augmentation to simulate interventions using only observational data.

- Second, we derive a simple algorithm that can select data augmentation techniques from a given list of transformations. We compare our approach to a variety of domain generalization methods on three domain generalization benchmarks. We demonstrate that we consistently outperform all other methods.

3.2 Method

3.2.1 Domain generalization

We first formalize the problem of domain generalization following the notations used in Muandet et al. [2013]. We assume that during training we have access to samples $\mathscr{S}$ from N different domains, where $\mathscr{S} = \{S^{d=i}\}_{i=1}^{N}$. From each domain n_i samples $S^{d=i} = \{(\mathbf{x}_k^{d=i}, y_k^{d=i})\}_{k=1}^{n_i}$ are included in the training set. The training data is represented as tuples of the form $(\mathbf{x}, y, d)$ sampled from the observational distribution $p(\mathbf{x}, y, d)$. The goal of domain generalization is to develop machine learning methods that generalize well to unseen domains. In order to test the ability of a machine learning model to generalize, we use samples $S^{d=N+1}$ from a previously unseen test domain $d = N+1$.

In this paper, we are interested in the general case where the observed domains d and targets y are spuriously correlated in the training dataset, i.e., where we might have $p(y|d=i) \neq p(y|d=j), i, j \in \{1,\dots,N\}$. Since the correlation between d and y is assumed to be spurious, it does not necessarily hold for the test domain $d = N+1$.

3.2.2 Domain generalization and data augmentation from a causal perspective

For readers unfamiliar with the concepts of causality, a brief introduction of the causal concepts that are used throughout this paper can be found in Appendix 3.6.7. For an in-depth introduction please see Pearl [2009] or Peters et al. [2017].

First, we introduce a Structural Causal Model (SCM) in order to describe what we believe in many cases reflects the underlying causal structure of domain generalization problems. The SCM is shown in Figure 3.1 (right)

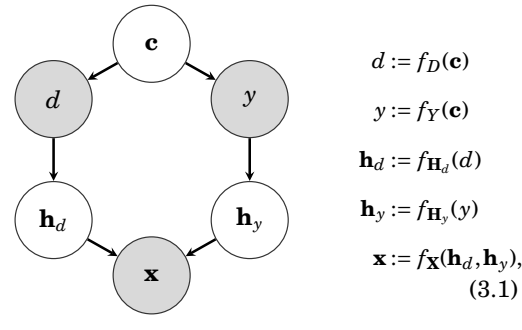


Figure 3.1. DAG and SCM with a hidden confounder.

where $\mathbf{c}$ is a hidden confounder (and a exogenous variable), d the domain,

y the target, $\mathbf{h}_d$ high-level features, e.g., color and orientation, caused by d, $\mathbf{h}_y$ high level-features, e.g., shape and texture, caused by y, and $\mathbf{x}$ the input. We omit including noise variables for clarity. The corresponding Directed Acyclic Graph (DAG) is shown in Figure 3.1 (left), where a grey node means the variable is observed and a white node corresponds to a latent (unobserved) variable. The presented DAG is similar to the ones constructed in Subbaswamy and Saria [2019] and Castro et al. [2020]. In Figure 3.1, the node $\mathbf{c}$ is a hidden confounder. The hidden confounder $\mathbf{c}$ opens up a backdoor path (a non-causal path) $d \leftarrow \mathbf{c} \rightarrow y$ [Pearl, 2009]. This path allows d to enter y trough the back door.

As a result, the domain d and the target y are in general no longer independent, $p(y,d) \neq p(y)p(d)$. Since the high-level features, $\mathbf{h}_d$ are children of d, they are spuriously correlated with y as well, i.e., $\mathbf{h}_d$ becomes predictive of y. We now assume that we train a machine learning model using ERM [Vapnik, 1992] and observational data generated from the DAG in Figure 3.1. The task is to predict y from $\mathbf{x}$, which itself is anti-causal. Since d and y are correlated, the machine learning model will likely rely on all high-level features $\mathbf{h}_d$ and $\mathbf{h}_y$ to predict y. Furthermore, we assume that the correlation of d and y is spurious. Therefore, it will not hold in general and will break under intervention. A machine learning model relying on high-level features $\mathbf{h}_d$ that are caused by d is thus likely to generalize poorly to unseen domains. Returning to our introductory example of classifying animals in images, the hidden confounder can be used to model the fact that there is a common cause for the type of animal and the landscape in an image. For example, the confounder could be the country where a particular image was taken, e.g., in Switzerland, we are more likely to see a cow standing in a green pasture than a camel or a desert.

3.2.3 Simulating interventions

One possible approach to deal with the spurious correlations between d and y is to intervene on d. Such an intervention would render d and y independent. In Figure 3.2 (left), we see the same DAG as in Figure 3.1 but after we intervened on d. We find that in Figure 3.2 (left) there is no more arrow connecting the hidden confounder $\mathbf{c}$ and the domain d. The backdoor path $d \leftarrow \mathbf{c} \rightarrow y$ has vanished. In the examples of animals and landscapes, we would have to physically move a cow to a desert to intervene on the domain d (the landscape). It becomes apparent that the interventions have to happen in the real world and are not operations on the already gathered observational data. In most domain generalization problems, it will not be feasible to collect new data with specific interventions.

In Figure 3.2 (center) we present a second way of addressing the problem of correlated variables d and y. In theory one could perform an intervention on all high-level features $\mathbf{h}_d$, i.e., $\text{do}(\mathbf{h}_d)$, since d affects $\mathbf{x}$ only indirectly

via $\mathbf{h}_d$, in our example $\mathbf{h}_d$ could represent the colors and textures of the landscapes. Again, an intervention like this would need to happen during the data collection process in the real world, e.g., by moving sand to a pasture.

However, we argue that in certain cases we can simulate data from the interventional distribution $p(\mathbf{x}, y|\text{do}(\mathbf{h}_d))$ using data augmentation in combination with observational data. For example, we could randomly perturb the colors in the animal images. This type of augmentation simulates a noise intervention on $\mathbf{h}_d$, i.e., $\text{do}(\mathbf{h}_d = \boldsymbol{\xi})$, where $\boldsymbol{\xi}$ is sampled from a noise distribution $N_{\boldsymbol{\xi}}$ [Peters et al., 2016].

In theory, we could intervene on $\mathbf{h}_d$ by setting $\mathbf{h}_d$ to a fixed value instead of performing a noise intervention. However, to simulate data from such an interventional distribution using data augmentation, we would require $\mathbf{h}_d$ to be observed, which we argue is generally not the case. In Appendix 3.6.10, we describe that there exist data augmentation methods that try to infer $\mathbf{h}_d$ for each sample $\mathbf{x}$ before setting $\mathbf{h}_d$ to a fixed value for all samples. Yet, these augmentations seem to perform worse than randomly sampled augmentations.

By augmenting only high-level features $\mathbf{h}_d$ that are caused by d we guarantee that the target y and features $\mathbf{h}_y$ are unchanged. After data augmentation the pairs $(\mathbf{x}_{\text{aug}}, y)$ should closely resemble samples from the interventional distribution $p(\mathbf{x}, y|\text{do}(\mathbf{h}_d))$. In Figure 3.2 (right) we see that we only require observational data from the DAG without any interventions. While each augmented sample $\mathbf{x}_{\text{aug}}$ individually can be seen as a counterfactual, we argue that we effectively marginalize over the counterfactual distribution by generating a multitude of augmented samples $\mathbf{x}_{\text{aug}}$ from each $\mathbf{x}$. We argue that for correctly chosen data augmentation we cannot distinguish the data generated by any of the three models in Figure 3.2.

If we want to choose data augmentation $\mathbf{x}_{\text{aug}} = \text{aug}(\mathbf{x})$, as a transformation $\text{aug}(\cdot)$ applied to observed data $\mathbf{x}$, such that it simulates an intervention on the high-level features $\mathbf{h}_d$ caused by d, one needs to make assumption about the causal data generating process. Formally, we require that augmenting the data $\mathbf{x}$ to $\mathbf{x}_{\text{aug}} = \text{aug}(\mathbf{x})$ commutes with an intervention $\text{do}(\mathbf{h}_d)$ prior to the data generation. We call this *intervention-augmentation equivariance*. In more formal detail, assume that we have the causal process from Equation 3.1: $\mathbf{x} := f_{\mathbf{X}}(\mathbf{h}_d, \mathbf{h}_y)$. Then augmenting $\mathbf{x}$ via $\text{aug}(\cdot)$ does:

$$\begin{aligned} \mathbf{x}_{\text{aug}} &= \text{aug}(\mathbf{x}) \\ &= \text{aug}(f_{\mathbf{X}}(\mathbf{h}_d, \mathbf{h}_y)). \end{aligned} \tag{3.2}$$

We then say that the causal process $f_{\mathbf{X}} : \mathscr{H}_d \times \mathscr{H}_y \mapsto \mathscr{X}$, is *intervention-augmentation equivariant* if for every considered stochastic data augmentation transformation $\text{aug}(\cdot)$ on $\mathbf{x} \in \mathscr{X}$ we have a corresponding noise inter-

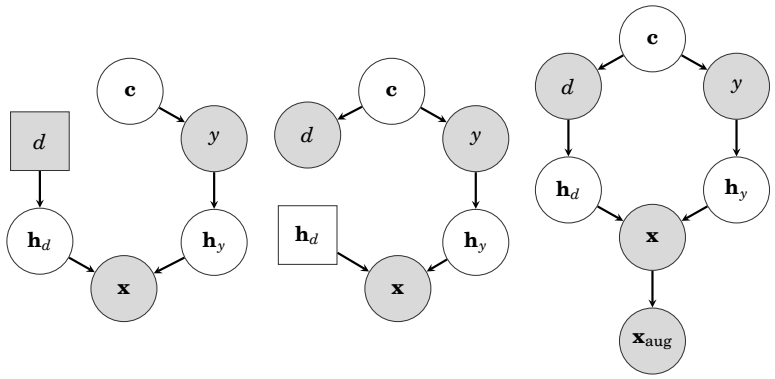


Figure 3.2. Left: DAG with hidden confounder after intervention on d. Center: DAG with hidden confounder after intervention on $\mathbf{h}_d$. Interventional nodes are squared. Right: DAG with hidden confounder plus data augmentation. Note that we do not have to intervene in the system that generates the data in the latter case. Data augmentation should be chosen in a way such that the augmented data simulates data from the center or left DAG.

vention $\text{do}(\cdot)$ on $\mathbf{h}_d \in \mathscr{H}_d$ such that:

$$\text{aug}(f_{\mathbf{X}}(\mathbf{h}_d, \mathbf{h}_y)) = f_{\mathbf{X}}(\text{do}(\mathbf{h}_d), \mathbf{h}_y). \tag{3.3}$$

The intervention-augmentation equivariance is expressed as a commutative diagram in Figure 3.3. We argue that by making strong assumptions about the true causal process we need to first identify the high-level features $\mathbf{h}_d$ caused by d. Second, we have to choose data augmentation $\text{aug}(\mathbf{x})$ that commutes with a corresponding intervention $\text{do}(\mathbf{h}_d)$ under the causal process $f_{\mathbf{X}}(\mathbf{h}_d, \mathbf{h}_y)$. A special case of intervention-augmentation equivari-

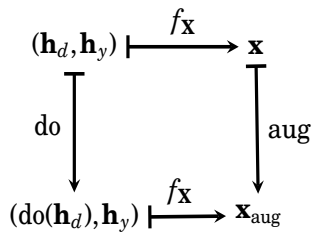


Figure 3.3. Intervention-augmentation equivariance expressed in a commutative diagram.

ance occurs in the classical case of an G-equivariant map $f_{\mathbf{X}}$, where G can be any (semi)group. For this to hold, we need G to act on the spaces $\mathscr{H}_y$, $\mathscr{H}_d$, $\mathscr{X}$, and we need to make sure that G acts trivially on $\mathscr{H}_y$. So any element $g \in G$ can transform elements $\mathbf{x} \in \mathscr{X}$ into $g \cdot \mathbf{x} \in \mathscr{X}$, which we will interpret as data augmentation, as demonstrated in Section 3.4. The elements $g \in G$ also transform $\mathbf{h}_d \in \mathscr{H}_d$ into $g \cdot \mathbf{h}_d \in \mathscr{H}_d$, which we consider

as a special type of intervention. Furthermore, $\mathbf{h}_y \in \mathscr{H}_y$ are assumed to be kept fixed $g \cdot \mathbf{h}_y = \mathbf{h}_y$ for all $g \in G$. So we put:

$$\text{do}(\mathbf{h}_d) := g \cdot \mathbf{h}_d, \tag{3.4}$$

$$\text{aug}(\mathbf{x}) := g \cdot \mathbf{x}, \tag{3.5}$$

where we assume that the elements $g \in G$ are randomly sampled from some distribution $p(g)$ on G. In this setting, any *G-equivariant map* $f_{\mathbf{X}}$ is then automatically also intervention-augmentation equivariant, as can be seen from:

$$\text{aug}(\mathbf{x}) = g \cdot f_{\mathbf{X}}(\mathbf{h}_d, \mathbf{h}_y) \tag{3.6}$$

$$= f_{\mathbf{X}}(g \cdot \mathbf{h}_d, g \cdot \mathbf{h}_y) \tag{3.7}$$

$$= f_{\mathbf{X}}(\text{do}(\mathbf{h}_d), \mathbf{h}_y), \tag{3.8}$$

a linear example of intervention-augmentation equivariance can be found in the Appendix.

In general, we find that the most frequently used data augmentations can be expressed as simple group actions. For example, randomly rotating the input image $\mathbf{x}$ can be understood as randomly sampling and applying elements g from the two-dimensional rotation group $SO(2)$ on the two-dimensional pixel grid. Randomly changing the hue of an image $\mathbf{x}$ corresponds to randomly sampling and applying elements g from the two-dimensional rotation group $SO(2)$, since hue can be represented as an angle in color space. Applying random permutations to the color channels of an image $\mathbf{x}$ is equivalent to randomly sampling and applying elements g from permutation group S_3, in the case of three separate color channels.

3.2.4 Selecting data augmentations for domain generalization

In Figure 3.2 (center), we see that if we successfully simulate an intervention on $\mathbf{h}_d$ using data augmentation the arrow from d to $\mathbf{h}_d$ vanishes. Based on this theoretical insight, we propose an algorithm that can select data augmentation techniques to improve domain generalization instead of manually choosing them. In the following, we will refer to the algorithm as Select Data Augmentation (SDA). Similar to Cubuk et al. [2019], we start with a list of data augmentation techniques including: 'brightness', 'contrast', 'saturation', 'hue', 'rotation', 'translate', 'scale', 'shear', 'vertical flip', and 'horizontal flip'. Since these transformations do not influence each other, they can be tested separately. The hyperparameter for each augmentation can be found in the Appendix. The proposed SDA algorithm consists of three steps:

1. We divide all samples from the training domains into a training and validation set.

2. We train a classifier to predict the domain d from input $\mathbf{x}$. During training, we apply the first data augmentation in our list to the training set samples. We save the domain accuracy on the validation set after training. We repeat this step with all data augmentations in the list.

3. We select the data augmentation with the lowest domain accuracy averaged over five seeds. If multiple data augmentations lie within the standard error of the selected one, they are selected as well, i.e., there is no statistically significant difference between the augmentations.

Intuitively, SDA will select data augmentation techniques that destroy information about d in $\mathbf{x}$. From a causal point of view, this is equivalent to weaken the arrow from d to $\mathbf{h}_d$. In Appendix 3.6.2, we perform an ablation study showing that SDA also reliably selects the most suitable data augmentation if the list contains the same augmentation with different hyperparameters.

There is one caveat, though. Throughout this entire section, we assume that we are successfully augmenting all high-level features $\mathbf{h}_d$ caused by d. In a real-world application, we usually have no means to validate this assumption. We might only augment a subset of $\mathbf{h}_d$. Furthermore, we might even augment high-level features $\mathbf{h}_y$ that are caused by the target node y. Nonetheless, we argue there are cases where we still obtain better generalization performance than a machine learning model trained without data augmentation. This may happen in cases where weakening the spurious confounding influence of $\mathbf{h}_d$ on y recovers more of the anti-causal signal for y than the data augmentation on the features $\mathbf{h}_y$ destroys. We evaluate this hypothesis empirically in Section 3.4.

3.3 Related work

3.3.1 Learning symmetries from data

In the previous section, we argue that choosing the right symmetry group for data augmentation relies on prior knowledge, e.g., preselecting a list of transformations to test. While this is a clear, practical limitation of our approach, there are no approaches that can learn symmetries from purely observational data to the best of our knowledge. Contemporary approaches like Lagrangian neural networks [Cranmer et al., 2020], graph neural networks [Kipf and Welling, 2017], and group-equivariant neural networks [Cohen and Welling, 2016] are enforcing apriori chosen symmetries instead of learning them.

3.3.2 Understanding data augmentation

Recently, Gontijo-Lopes et al. [2020] develop two measures: affinity and diversity. The measures are used to quantify the effectiveness of existing data augmentation methods. They find that augmentations that have high affinity and diversity scores lead to better generalization performance. While affinity and diversity rely on the iid assumption, we provide an alternative for non-iid datasets. Lyle et al. [2020] investigate how data augmentation can be used to incorporate invariance into machine learning models. They show that while data augmentation can lead to tighter PAC-Bayes bounds, data augmentation is not guaranteed to lead to invariance. In Equation, 3.3 we formalize under which condition (namely intervention-augmentation equivariance) data augmentation will lead to invariance.

3.3.3 Advanced data augmentation techniques

Zhang et al. [2018] introduced a method called mixup that constructs new training examples by linearly interpolating between two existing examples $(\mathbf{x}_i,\ y_i)$ and $(\mathbf{x}_j,\ y_j)$. In Gowal et al. [2020] and Perez and Wang [2017] a Generative Adversarial Network (GAN) is used to perform so-called 'adversarial mixing'. The GAN can generate new training examples that belong to the same class y but have different styles. Furthermore, Perez and Wang [2017] propose a novel method called 'neural augmentation' where they train the first part of their model to generate an augmented image from two training examples with the same class y.

3.3.4 Causality

In Peters et al. [2016] a method for Invariant Causal Prediction (ICP) is developed. It is built on the assumption that causal features are stable given different experimental settings. Given the complete set of causal features, the conditional distribution of the target variable y must remain the same under interventions, e.g., change of the domain. Whereas predictions made by a machine learning model relying on non-causal features are generally not stable under interventions. Recently, Arjovsky et al. [2020] proposed a framework called Invariant Risk Minimization (IRM) that shares the same goal as ICP. In IRM, a soft penalty in combination with an ERM term is used to balance the invariance and predictive power of the learned machine learning model. In contrast to ICP, IRM can be used for tasks on unstructured data, e.g., images. However, while both methods (ICP and IRM) try to learn features that are parents of y, we argue that for the majority of domain generalization problems, the task of predicting y from $\mathbf{x}$ is anti-causal. Therefore we are interested in augmenting only features caused by d, i.e., the descendants of d, assuming that the

remaining features are caused by y. In Arjovsky et al. [2020], they argue that there exists a discrepancy between the true label (part of the true causal mechanism) that caused $\mathbf{x}$ and the annotation produced by human labelers. Learning this 'labeler function' will lead to a good generalization performance, even though it might rely on patterns that are anti-causal or non-causal. In this situation, the IRM objective becomes ineffective.

Heinze-Deml and Meinshausen [2021] introduced the Conditional variance Regularization (CoRe). CoRe uses grouped observations (e.g., training samples with the same class y but different styles) to learn invariant representations. Samples are grouped by an additional ID variable, which is different from the label y. We find that in most cases, it is difficult to obtain an additional ID variable, e.g., none of the datasets in Section 3.4 features such a variable. If no such ID variable exists, CoRe can use pairs of original images and augmented images to learn invariant representations.

While we are focusing on the DAG in Figure 3.1, Bareinboim and Pearl [2016] and Mooij et al. [2020] have developed general graphical representations for relating data generating processes across domains. If the confounder $\mathbf{c}$ was observed methods that find stable feature sets such as those in Rojas-Carulla et al. [2018] and Magliacane et al. [2018], could be used. Furthermore, Subbaswamy et al. [2019] shows that instead of intervening in some cases, it is possible to fit an interventional distribution from observational data. However, imaging data poses a challenge that existing causal-based methods cannot deal with, thus motivating data augmentation.

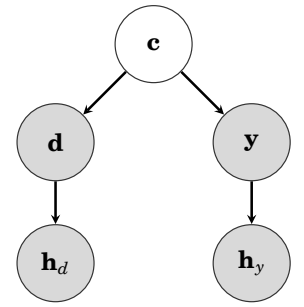
3.4 Experiments

We evaluate the performance of data augmentation in combination with Empirical Risk Minimization (ERM) [Vapnik, 1992] on four datasets. While the first is a synthetic dataset, the other three are benchmark image datasets (rotated MNIST, colored MNIST, and PACS) where the domain d and target y are confounded. The synthetic dataset is used to study the effect of data augmentation on a model's performance when high level-features caused by domain and high level-features caused by the label are augmented. For the benchmark image datasets, we first use SDA to select the best data augmentation techniques. The results for this first step can be found in Table 3.5 in the Appendix. Afterward, we apply the selected data augmentations and train the respective model using ERM. Finally, we perform an ablation study that applies all data augmentations to all three image datasets instead of the selected ones.

Code to replicate all experiments can be found under `https://github.com/AMLab-Amsterdam/DataAugmentationInterventions`.

3.4.1 Synthetic data

For the first experiment we simulate data from the linear Gaussian SCM in Figure 3.4 (right), where the corresponding DAG can be seen in Figure 3.4 (left).



$$
\begin{aligned}
\mathbf{c} &:= \mathcal{N}(0, \sigma_{\mathbf{c}}^2) \\
\mathbf{d} &:= \mathbf{c} \cdot W_{\mathbf{c}\to\mathbf{d}} + \mathcal{N}(0, \sigma^2) \\
\mathbf{y} &:= \mathbf{c} \cdot W_{\mathbf{c}\to\mathbf{y}} + \mathcal{N}(0, \sigma^2) \\
\mathbf{h}_d &:= \mathbf{d} \cdot W_{\mathbf{d}\to\mathbf{h}_d} + \mathcal{N}(0, \sigma^2) \\
\mathbf{h}_y &:= \mathbf{y} \cdot W_{\mathbf{y}\to\mathbf{h}_y} + \mathcal{N}(0, \sigma^2),
\end{aligned} \tag{3.9}
$$

Figure 3.4. DAG and linear Gaussian SCM for synthetic data.

We choose $\mathbf{c}$, $\mathbf{d}$, $\mathbf{y}$, $\mathbf{h}_d$ and $\mathbf{h}_y$ to be five dimensional vectors. Furthermore, we sample the elements of the square matrices $W_{\mathbf{c}\to\mathbf{d}}$, $W_{\mathbf{c}\to\mathbf{y}}$, $W_{\mathbf{d}\to\mathbf{h}_d}$ and $W_{\mathbf{y}\to\mathbf{h}_y}$ from $\mathcal{N}(0, I)$. In all of our experiments $\sigma_{\mathbf{c}} = I$ and $\sigma = 0.1 \cdot I$. The task is to regress $\sum_i^5 y_i$ from $\mathbf{x}$, where $\mathbf{x} = [\mathbf{h}_d, \mathbf{h}_y]$, a 10 dimensional feature vector. During training the data is generated using the DAG in Figure 3.4 (left), where due the confounder $\mathbf{c}$ the features $\mathbf{h}_d$ and $\mathbf{y}$ are spuriously correlated. During testing we set $\mathbf{d} := \mathcal{N}(0, I)$, keeping $W_{\mathbf{c}\to\mathbf{d}}$, $W_{\mathbf{c}\to\mathbf{y}}$, $W_{\mathbf{d}\to\mathbf{h}_d}$ and $W_{\mathbf{y}\to\mathbf{h}_y}$ the same as during training. As a result, features $\mathbf{h}_d$ and $\mathbf{y}$ are no longer correlated. A model relying on features $\mathbf{h}_d$ will not be able to generalize well to the test data. In all experiments, we use linear regression to minimize the empirical risk. As our data augmentation technique, we choose to add noise sampled from a uniform distribution $U[-10, 10]$. We vary the number of dimensions of $\mathbf{h}_d$ as well as of $\mathbf{h}_y$ that are augmented. Each experiment is repeated 50 times. In Figure 3.5 we plot the mean of the mean squared error (MSE) together with the standard error.

In Figure 3.5, we see that ERM using only features $\mathbf{h}_y$ (pink line) achieves the lowest MSE. Next, we apply data augmentation to one, two, three, four, and five dimensions of $\mathbf{h}_d$ while keeping $\mathbf{h}_y$ unchanged (orange line). We find that if data augmentation is applied to all five dimensions of $\mathbf{h}_d$, we can match the MSE of ERM with only features $\mathbf{h}_y$. In this case, we satisfy the condition in Equation 3.3. Furthermore, we find that unsurprisingly the MSE of models trained with data augmentation applied to features $\mathbf{h}_y$ increases (green, red, purple, and brown line). However, we can see that as long as we apply data augmentation to at least three dimensions of $\mathbf{h}_d$, the resulting MSE is lower than ERM using all features $\mathbf{h}_d$ and $\mathbf{h}_y$ (blue line). Perhaps the most surprising result of this experiment is that there exist conditions under which applying data augmentation to features caused by

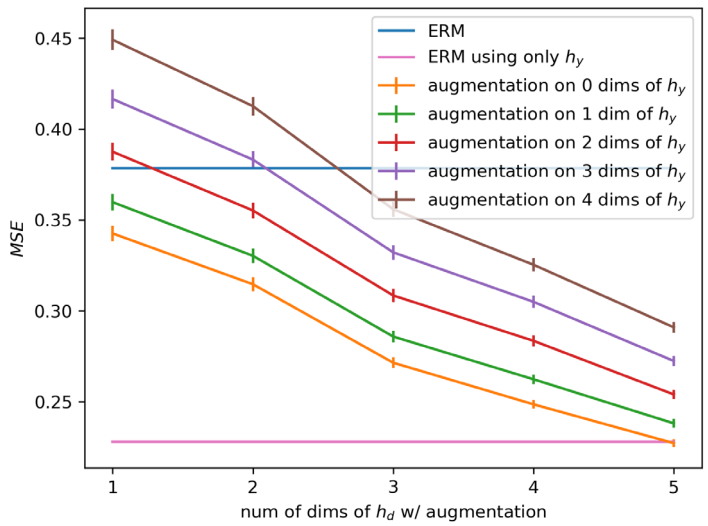


Figure 3.5. Results on synthetic data.

d *and* features caused by **y** will result in better generalization performance compared to ERM using all features.

3.4.2 Rotated MNIST

We construct the rotated MNIST dataset following Li et al. [2018b]. This dataset consists of four different domains d and ten different classes y, each domain corresponds to a different rotation angle: $d = \{0°, 30°, 60°, 90°\}$. We first randomly select a subset of images $\mathbf{x}$ from the MNIST training dataset and afterward apply the rotation to each image of the subset. For the next domain, we randomly select a new subset. To guarantee the variance of $p(y)$ among the domains, the number of training examples for each digit class y is randomly chosen from a uniform distribution $U[80, 160]$.

For each experiment, three domains are selected for training, and one domain is selected for testing. For the test domain, the corresponding rotation is applied to the 10000 examples of the MNIST testset. In Table 3.2, we compare data augmentation in combination with ERM to ERM, a Domain Adversarial Neural Network (DANN) [Ganin et al., 2016] and a Conditional Domain Adversarial Neural Network (CDANN) [Li et al., 2018b]. All methods use a LeNet [LeCun et al., 1998] type architecture and we repeat each experiment 10 times. First, we use SDA to find the best data augmentation technique, where we use the same LeNet model and training procedure for the domain classifier and only samples from the training domains. The data augmentation with the lowest domain accuracy in all four cases, where we leave out one of the domains for testing,

is 'rotation'. In addition, we perform an ablation study showing that SDA reliably picks the most suitable hyperparameters. The results can be found in Table 3.4 in the Appendix. Second, we apply random rotations between 0° and 359° to the images $\mathbf{x}$ during training, denoted by DA. If we assume $\mathbf{h}_d$ to be equal to the rotation angle of the MNIST digit in a given image $\mathbf{x}$, applying random rotations to $\mathbf{x}$ is equal to a noise intervention on $\mathbf{h}_d$, see Equation 3.3. As described in Section 3.2, applying random rotations to $\mathbf{x}$ can be understood as randomly sampling elements g from the two-dimensional rotation group $SO(2)$. Note that elements $g \in SO(2)$ act trivially on $\mathbf{h}_y$: Rotations do not change the digit shapes. The result is a training dataset where d and y are independent. In Table 3.2, we see that the results of DA are similar for all four test domains. Furthermore, we find that DA outperforms ERM, DANN, and CDANN, where CDANN is specially designed for the case where d and y are spuriously correlated.

Table 3.1. Results on Colored MNIST. Average accuracy ± standard deviation for ten seeds.

Acc	ERM	IRM	REx	SDA
Train	**87.4 ± 0.2**	70.8 ± 0.9	71.5 ± 1.0	72.1 ± 0.4
Test	17.1 ± 0.6	66.9 ± 2.5	68.7 ± 0.9	**74.1 ± 0.9**

Table 3.2. Results on Rotated MNIST results. Average accuracy for ten seeds.

Target	ERM	DANN	CDANN	SDA
0°	75.4	77.1	78.5	**96.1**
30°	93.4	94.2	94.9	**95.9**
60°	94.5	95.2	95.6	**95.7**
90°	79.6	83.0	84.0	**95.9**
Ave	85.7	87.4	88.3	**95.9**

Table 3.3. Results on PACS dataset. Average accuracy for five seeds.

Test	ERM	CDANN	L2G	GLCM	SSN	IRM	REx	MetaReg	JigSaw	SDA
A	63.3	62.7	66.2	66.8	64.1	67.1	67.0	69.8	67.6	**70.45**
C	63.1	69.7	66.9	69.7	66.8	68.5	68.0	70.4	**71.7**	68.49
P	87.7	78.7	88.0	87.9	90.2	89.4	89.7	**91.1**	89.0	88.35
S	54.1	64.5	59.0	56.3	60.1	57.8	59.8	59.3	65.2	**72.24**
Ave	67.1	68.9	70.0	70.2	70.3	70.7	71.1	72.6	73.4	**74.9**

3.4.3 Colored MNIST

Following Arjovsky et al. [2020], we create a version of the MNIST dataset where the color of each digit is spuriously correlated with a binary label y. We construct two training domains and one test domain where the digits

of the original MNIST classes '0' to '4' are labeled $y = 0$, and the digits of the classes '5' to '9' are labeled $y = 1$. Subsequently, for 25% of the digits, we flip the label y. We now color digits labeled $y = 0$ red and digits labeled $y = 1$ green. Last, we flip the color of a digit with a probability of 0.2 for the first training domain and 0.1 for the second training domain. In the test domain, the color of a digit is flipped with a probability of 0.9. By design, the original MNIST class of each digit ('0' to '9') is a direct cause of the new label y, whereas the color of each digit is a descendant of the new label y.

The DAG of the colored MNIST, shown in Appendix Figure 3.6, deviates slightly from the DAG in Figure 3.1, nonetheless the reasoning in Section 3.2 is still valid. In Table 3.1, we see that while ERM is performing well on the training domains, it fails to generalize to the test domain since it is using the color information to predict y. In contrast, IRM [Arjovsky et al., 2020] and REx [Krueger et al., 2020] generalizes well to the test domain. Again, we use SDA to find the appropriate data augmentations. We use the same MLP and training procedure as in Arjovsky et al. [2020] for the domain classifier. We want to highlight that SDA only relies on samples from the two training domains, whereas the hyperparameters of IRM and REx were fine-tuned on samples from the test domain as described in Krueger et al. [2020]. In the case of the colored MNIST dataset, the selected data augmentations are 'hue' and 'translate', denoted by DA. As described in Section 3.2, applying random permutations to the hue value of $\mathbf{x}$ is equivalent to randomly sampling and applying elements g from the permutation group $SO(2)$. We argue that elements g do not change $\mathbf{h}_y$: high-level features that contain information about the shape of each digit. In our experiment, we use the same network architecture and training procedure as described in Arjovsky et al. [2020]. Each experiment is repeated ten times. We find that DA can successfully weaken the spurious confounding influence of the domain d on y, see Table 3.1.

3.4.4 PACS

The PACS dataset [Li et al., 2017] was introduced as a strong benchmark dataset for domain generalization methods that features large domain shifts. The dataset consists of four domains: d = ['photo' (P), 'art-painting' (A), 'cartoon' (C), 'sketch' (S)], i.e., each image style is viewed as a domain. The numbers of images in each domain are 1670, 2048, 2344, 3929, respectively. There are seven classes: y = [dog, elephant, giraffe, guitar, horse, house, person]. We fine-tune an AlexNet-model [Krizhevsky et al., 2012], that was pre-trained on ImageNet, using ERM in combination with data augmentation. We apply SDA to select the data augmentation for the following experiment. For the domain classifier, we fine-tune an AlexNet-model as described above. In addition, we use a cross-validation procedure where we leave one domain out and use the three domains for

training. SDA determines four data augmentation techniques to be useful: 'brightness', 'contrast', 'saturation', and 'hue'. In combination, these four augmentations are commonly called color jitter or color perturbations. By randomly applying color perturbations, we are weakening the spurious confounding influence of $\mathbf{h}_d$ on y, as described in Section 3.2. In Table 3.3, we compare DA to various domain generalization methods: CDANN [Li et al., 2018b], L2G [Li et al., 2018a], GLCM [Wang et al., 2018], SSN [Mancini et al., 2018], IRM [Arjovsky et al., 2020], REx [Krueger et al., 2020], MetaReg [Balaji et al., 2018], JigSaw [Carlucci et al., 2019a], where all methods use the same pre-trained AlexNet-model. We repeat each experiment 5 times and report the average accuracy. We find that DA obtains the highest average accuracy. The biggest performance gains of DA compared to ERM are on the test domains 'art painting' and 'sketch'. For example, the domain 'sketch' consists of black sketches of the seven object classes on white background, see Figure 3.7. Since the object's color is not correlated with the class, a model relying on color features will generalize poorly to the 'sketch' domain. However, by randomly changing the colors of the images in the training domains ('art painting', 'cartoon', 'photo'), we find that DA can generalize much better.

Ablation study: Using all data augmentation techniques We repeat the previous experiments on Rotated MNIST, Colored MNIST, and PACS using all data augmentation techniques listed in the Appendix. We compare the accuracy of a classifier trained using all data augmentation techniques to a classifier trained using SDA. We find that using all data augmentation techniques together results in a significant drop in performance for all three datasets: 25.4% for Rotated MNIST, 8.7% for Colored MNIST, and 16.1% for PACS. We observe combinations of datasets and data augmentation techniques that lead to a drastic drop in performance on their own, e.g., the PACS dataset and random rotations. We argue that a model trained without random rotations exploits the fact that, e.g., the orientation of an animal or person is usually upright. This example shows that we cannot simply describe data augmentation as 'label-preserving transformations' since a rotated animal or person will still have the same label.

3.5 Conclusion

In this paper, we present a causal perspective on the effectiveness of data augmentation in the context of domain generalization. Using an SCM, we address a core problem of domain generalization: the spurious correlation of the domain variable d and the target variable y. While, in theory, we could intervene on the domain variable d, this solution is impractical since we assume that we only have access to observational data. However, we show that data augmentation can be a surrogate tool for simulating in-

terventions on the domain variable d and its children. Prior knowledge can be used to choose data augmentation techniques that only act on the non-descendants of the target variable y. Furthermore, we show that randomly applying data augmentation can be understood as randomly sampling elements from common symmetry groups. In addition, we propose a simple algorithm to select suitable augmentation techniques from a given list of transformations. We use a domain classifier to measure how well each augmentation can weaken the causal link between the domain d and $\mathbf{h}_d$ high-level features caused by d. We evaluated this approach on four different datasets and were able to show that empirical risk minimization in combination with accurately selected data augmentation results in good generalization performance. The analysis in this paper could be further used to design data augmentation to simulate interventional datasets for domain generalization methods by exploiting intervention-augmentation equivariance.

3.6 Appendix

3.6.1 Additional details for SDA

All data augmentations are implemented using the `TORCHVISION.TRANSFORMS` module of PyTorch [Paszke et al., 2019]. We choose the range of the hyperparameters of the augmentations in such a way that they do not destroy all information in $\mathbf{x}$, e.g., setting the brightness of all pixels to 0 or translating all pixels by the full image width. In all experiments we use the following data augmentations:

- `'brightness'`:
  `torchvision.transforms.ColorJitter(brightness=1.0, contrast=0, saturation=0, hue=0)`
- `'contrast'`:
  `torchvision.transforms.ColorJitter(brightness=0, contrast=10.0, saturation=0, hue=0)`
- `'saturation'`:
  `torchvision.transforms.ColorJitter(brightness=0, contrast=0, saturation=10.0, hue=0)`
- `'hue'`:
  `torchvision.transforms.ColorJitter(brightness=0, contrast=0, saturation=0, hue=0.5)`
- `'rotation'`:
  `torchvision.transforms.RandomAffine([0, 359], translate=None, scale=None, shear=None, resample=PIL.Image.BILINEAR, fillcolor=0)`
- `'translate'`:
  `torchvision.transforms.RandomAffine(0, translate=[0.2, 0.2], scale=None, shear=None, resample=PIL.Image.BILINEAR, fillcolor=0)`
- `'scale'`:
  `torchvision.transforms.RandomAffine(0, translate=None, scale=[0.8, 1.2], shear=None, resample=PIL.Image.BILINEAR, fillcolor=0)`
- `'shear'`:
  `torchvision.transforms.RandomAffine(0, translate=None, scale=None, shear=[-10., 10., -10., 10.], resample=PIL.Image.BILINEAR, fillcolor=0)`
- `'vflip'`:
  `torchvision.transforms.RandomVerticalFlip(p=0.5)`
- `'hflip'`:
  `torchvision.transforms.RandomHorizontalFlip(p=0.5)`

3.6.2 Ablation study on rotated MNIST

We will demonstrate now that SDA can also be used to find the most suitable hyperparameters for the data augmentations used in this paper. In this example, we focus on the rotated MNIST dataset and the data augmentation 'rotate'. We use the same experimental setup as described in the rotated MNIST experiment. We choose $\{30°, 60°, 90°\}$ as training domains and $0°$ as the test domain. We compare five sets of hyperparameters, where each set defines the range from which the rotation angle is uniformly sampled. In Table 3.4, we find that the hyperparameters $[0°, 359°]$ lead to the lowest domain accuracy, i.e., simulate an intervention on $\mathbf{h}_d$ the best.

Table 3.4. Comparing domain accuracy on rotated MNIST for five different sets of the data augmentation 'rotate'. Average ± standard error over five seeds.

Hyperparameter	domain accuracy
$[-15°, 15°]$	92.60 ± 0.98
$[-45°, 45°]$	82.63 ± 0.89
$[-90°, 90°]$	69.79 ± 0.91
$[0°, 180°]$	63.16 ± 1.51
$[0°, 359°]$	51.70 ± 2.21

3.6.3 Results of domain classifier on each dataset

We train a domain classifier for each dataset using the same architecture and training procedure as used for the label classifier. We only use samples from the training domains and repeat each experiment five times. In Table 3.5, we show the domain accuracy for each of the datasets. In the

case of rotated MNIST, we perform four experiments where each of the domains $d = \{0°, 30°, 60°, 90°\}$ is used for testing once, while the remaining three domains are used for training. For each experiment, SDA returns the augmentation 'rotate' as the most suitable. In Table 3.5, we show the average of the four experiments that were each repeated five times. In the case of colored MNIST, the training and test domains are fixed. Therefore, we only conducted one experiment. We show the average of the one experiment that was repeated five times. For PACS, we perform four experiments where each of the domains $d = \{$'photo', 'art painting', 'cartoon', 'sketch'$\}$ is used for testing once, while the remaining three domains are used for training. We use cross-validation over all four experiments to select the data augmentation. In Table 3.5, we show the average of the four experiments that were each repeated five times.

Table 3.5. Domain accuracy for each dataset. Average ± standard error.

Data Augmentation	rotated MNIST	Colored MNIST	PACS
'brightness'	98.45 ± 0.24	50.1524 ± 0.1527	96.46 ± 0.37
'contrast'	98.64 ± 0.23	50.1470 ± 0.0506	96.41 ± 0.37
'saturation'	98.95 ± 0.21	50.1894 ± 0.0593	96.03 ± 0.43
'hue'	98.66 ± 0.36	50.0006 ± 0.0028	96.32 ± 0.41
'rotation'	64.70 ± 2.21	50.0024 ± 0.0030	96.59 ± 0.39
'translation'	90.84 ± 1.65	50.0004 ± 0.0008	96.82 ± 0.34
'scale'	91.42 ± 1.34	50.2082 ± 0.1327	97.00 ± 0.29
'shear'	91.48 ± 1.14	50.2252 ± 0.1531	96.82 ± 0.34
'vertical flip'	88.79 ± 0.50	50.1560 ± 0.0140	96.88 ± 0.34
'horizontal flip'	91.98 ± 0.29	50.4060 ± 0.0274	96.54 ± 0.33

3.6.4 Colored MNIST

The DAG of the data generating process for the colored MNIST experiment is shown in Figure 3.6 (left), where d is the domain, y is the binary label, $\hat{y}$ is the original MNIST label, $\mathbf{h}_d$ are high-level color features caused by d and y, $\mathbf{h}_y$ are high-level shape features caused by $\hat{y}$, and $\mathbf{x}$ is the observed image. In the case of the colored MNIST dataset, the spurious correlation between d and y is the result of the collider $\mathbf{h}_d$ (that itself is a parent of the observed node $\mathbf{x}$). While the cause of the spurious correlation between d and y is different, the reasoning in Section 3.2 is still valid. In Figure 3.6 (right), we show that in theory an intervention on $\mathbf{h}_d$ will remove the spurious correlation between d and y. We argue that an intervention on $\mathbf{h}_d$ can be simulated by data augmentation. We present experimental evidence in Section 3.4.

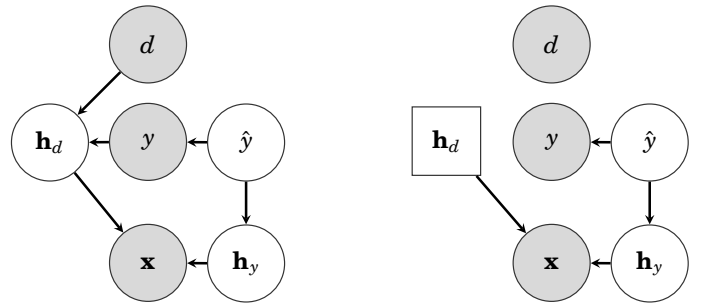


Figure 3.6. Left: DAG of the data generating process for the colored MNIST dataset. Right: The same DAG after intervention on $\mathbf{h}_d$. Interventional nodes are squared.

3.6.5 PACS

Example images of the PACS dataset, see Figure 3.7

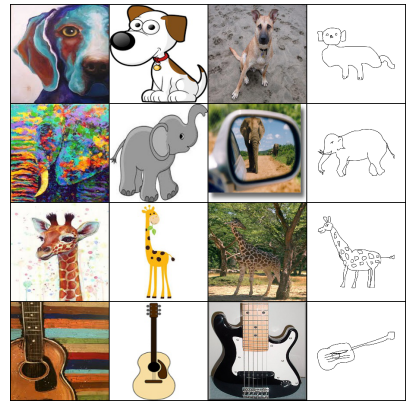


Figure 3.7. Samples from the first four classes ('dog', 'elephant', 'giraffe', 'guitar') for each domain (art-painting (A), cartoon (C), photo (P), sketch (S)) of the PACS dataset [Li et al., 2017].

3.6.6 Linear example of intervention-augmentation equivariance

A simple linear example can be constructed where the domain d causes a specific ordering in $\mathbf{h}_d$ that is spuriously correlated with the label y. In addition, G is the permutation group and $g \in G$ acts as a permutation matrix A on $\mathbf{x}$, i.e., $A\mathbf{x} = g \cdot \mathbf{x}$. In particular, we assume that $f_{\mathbf{X}}(\cdot)$ is a linear transformation

$$\mathbf{x} = f_{\mathbf{X}}(\mathbf{h}_d, \mathbf{h}_y) = C\mathbf{h}_d + D\mathbf{h}_y + \mathbf{e}, \tag{3.10}$$

where $\mathbf{x}, \mathbf{h}_d, \mathbf{h}_y, \mathbf{e}$ are vectors and C, D are matrices correspondingly sized. The data augmentation can be expressed as a linear transformation of the form

$$\mathbf{x}_{\text{aug}} = A\mathbf{x}, \tag{3.11}$$

where A is a correspondingly sized matrix sampled from the set of all permutation matrices. Combining Equation 3.10 and 3.11, we obtain

$$\begin{aligned}\mathbf{x}_{\text{aug}} &= A\mathbf{x} \\ &= AC\mathbf{h}_d + AD\mathbf{h}_y + A\mathbf{e} \\ &= C\left(C^{-1}AC\mathbf{h}_d\right) + AD\mathbf{h}_y + A\mathbf{e} \\ &= f_{\mathbf{X}}\left(\text{do}_A(\mathbf{h}_d), \mathbf{h}_y\right).\end{aligned} \tag{3.12}$$

We find that if that $AD = D$ and $A\mathbf{e} = \mathbf{e}$, i.e., D and $\mathbf{e}$ are permutation invariant, the transformation $A\mathbf{x} = g \cdot \mathbf{x}$ successfully simulates the noise intervention $\text{do}_A(\mathbf{h}_d) := C^{-1}AC\mathbf{h}_d$ (with slight abuse of notation), i.e., we find that it satisfy the intervention-augmentation equivariance condition.

3.6.7 Causality

What follows is a brief introduction of causal concepts that are used throughout this paper. It hopefully makes the paper more self-contained and more accessible for readers who encounter these concepts for the first time. For an in-depth introduction please see Pearl [2009] or Peters et al. [2017].

Structural causal models We say that a set of variables $x_1, \ldots, x_l$ causes a variable y if *intervening* on any of the x_m changes the distribution of y. This is usually different from (conditional) *observational* dependence between the x_m and y. Structural Causal Models (SCMs) are used to formalize those causal interactions between variables. We need to distinguish between two types of variables: exogenous and endogenous variables. Exogenous variables, usually unobserved independent random variables, can be seen as an entry point to our SCM. The endogenous variables x_m are then determined by the causal mechanisms, which are formalized via functional relations: $x_m = f_m(x_{\text{pa}_m})$, where x_{pa_m} is the tuple of the so-called parent variables of x_m. These relations of an SCM induce a corresponding graphical model. This paper only deals with acyclic relationships, leading to Directed Acyclic Graphs (DAGs) as part of a Bayesian network. In Figure 3.8, we see three SCMs and their corresponding DAGs. Note that the direction of the arrows indicates the causal direction.

The SCMs in Figure 3.8 are considered to be the three main building blocks of every causal model: chain, confounder, and collider. Where each

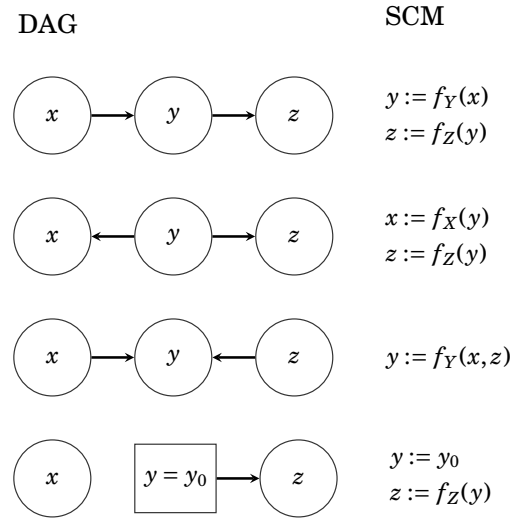


Figure 3.8. Causal structures. Top to bottom: chain, confounder, collider, chain with intervention on y.

of them introduces a different (conditional in-)dependence structure. First row: In case of a chain the variables x and z become conditionally independent if we condition on the center variable y, i.e., $p(z|x,y) = p(z|y)$. Second row: An observed confounder y can introduce spurious correlation between its two children variables x and z, i.e., we may have $p(x,z) \neq p(x)p(z)$. If we condition on the confounding variable y they become conditionally independent again, i.e., $p(z|x,y) = p(z|y)$ and $p(x|z,y) = p(x|y)$. Third row: In case of an unobserved collider y the two parent variables are independent, $p(x,z) = p(x)p(z)$. However, if we condition on y they may become conditionally dependent, i.e., $p(x,z|y) \neq p(x|y)p(z|y)$.

Interventions In its simplest form an intervention can be described as setting a variable y to a constant value, e.g., $y = y_0$ irrespective of its parent variables. The result of such an intervention on the SCM of a chain and the corresponding DAG can be seen in the bottom row of Figure 3.8. In this example, the variable y becomes independent of its parent variable x, i.e., we are replacing the function assignment $y = f(x)$ with $y = y_0$, effectively deleting the function $f(\cdot)$ and the corresponding arrow in the DAG. Using the do-*operator* [Pearl, 2009] we can write the resulting interventional distribution as follows: $p(z|x, \text{do}(y = y_0)) = p(z|\text{do}(y = y_0))$. In this paper, we use a special form of interventions, so-called noise or stochastic interventions [Peters et al., 2016]. Instead of setting the intervened variable to a fixed value, we randomize the values of y, i.e., $\text{do}(y = \xi)$, where ξ is sampled from a noise distribution N_ξ.

3.6.8 Domain generalization

Arguably, the most commonly used approach in domain generalization relies on learning domain invariant features. The learning of invariant features can be achieved by mapping an input $\mathbf{x}$ to intermediate features $\mathbf{z}$ that are uninformative of the domain d, i.e., $p(\mathbf{z}|d=i) = p(\mathbf{z}|d=j)$. At the same time, the intermediate features $\mathbf{z}$ are optimized for a low prediction error on all training domains. This results in finding a saddle point for the setting commonly referred to as domain adversarial learning [Ganin et al., 2016]. It is assumed that such $\mathbf{z}$ will generalize well to the test domain and, thus, result in a low test error.

Recent work of Zhao et al. [2019], Johansson et al. [2019] and Arjovsky et al. [2020], in the context of domain adaptation, shows that enforcing $p(\mathbf{z}|d=i) = p(\mathbf{z}|d=j)$ is not necessarily leading to a low test error if the domains d and targets y are spuriously correlated, i.e., $p(y|d=i) \neq p(y|d=j)$. We now extend the findings of Zhao et al. [2019] to domain generalization.

As shown in Zhao et al. [2019], an information-theoretic lower bound can be derived for the domain adaptation case. The bound "demonstrates that learning invariant representations could lead to a feature space where the joint error on both domains is large." We provide a straightforward extension of this bound for the domain generalization case.

Introduction of notation:

- $\mathbf{x}$: input
- $\mathbf{z}$: intermediate representation
- $\hat{y}$: output
- function composition: $\mathbf{x} \xrightarrow{g} \mathbf{z} \xrightarrow{h} \hat{y}$
- y: true label
- h: function mapping $\mathbf{x}$ to $\mathbf{z}$
- g: function mapping $\mathbf{z}$ to $\hat{y}$
- JSD: Jensen-Shannon divergence
- $\epsilon^{d=i}$: empirical risk on domain $d=i$

Besides, we need the following two lemmas from Zhao et al. [2019]. Proofs can be found in Zhao et al. [2019].

Lemma 4.6:

$$\mathrm{JSD}(p(\hat{y}|d=i)||p(\hat{y}|d=j)) \tag{3.13}$$

$$\leq \mathrm{JSD}(p(\mathbf{z}|d=i)||p(\mathbf{z}|d=j)), \tag{3.14}$$

where $p(\hat{y}|d=i)$ are the marginal distributions of the output in domain $d=i$ and $p(\mathbf{z}|d=i)$ are the marginal distributions of the intermediate representation in domain $d=i$.

Lemma 4.7:

$$\mathrm{JSD}(p(y|d=i)||p(\hat{y}|d=i)) \leq \sqrt{\epsilon_i(h \circ g)}, \tag{3.15}$$

i.e., how well is my output distribution matching the true labels distribution.

We start with the pairwise sum of Jensen-Shannon divergence between all N training domains and the $N+1$ test domain

$$\sum_{1 \leq i < j \leq N+1} \mathrm{JSD}(p(y|d=i)||p(y|d=j)). \tag{3.16}$$

Since JSD is a metric we can write

$$\sum_{1 \leq i < j \leq N+1} \mathrm{JSD}(p(y|d=i)||p(y|d=j)) \tag{3.17}$$

$$\leq \sum_{1 \leq i < j \leq N+1} \mathrm{JSD}(p(\hat{y}|d=i)||p(\hat{y}|d=j)) \tag{3.18}$$

$$+2\sum_{k}^{N+1} \mathrm{JSD}(p(y|d=k)||p(\hat{y}|d=k)). \tag{3.19}$$

Using Lemma 4.6 we get

$$\sum_{1 \leq i < j \leq N+1} \mathrm{JSD}(p(y|d=i)||p(y|d=j)) \tag{3.20}$$

$$\leq \sum_{1 \leq i < j \leq N+1} \mathrm{JSD}(p(\mathbf{z}|d=i)||p(\mathbf{z}|d=j)) \tag{3.21}$$

$$+2\sum_{k}^{N+1} \mathrm{JSD}(p(y|d=k)||p(\hat{y}|d=k)). \tag{3.22}$$

Using Lemma 4.7 we get

$$\sum_{1 \leq i < j \leq N+1} \mathrm{JSD}(p(y|d=i)||p(y|d=j)) \tag{3.23}$$

$$\leq \sum_{1 \leq i < j \leq N+1} \mathrm{JSD}(p(\mathbf{z}|d=i)||p(\mathbf{z}|d=j)) \tag{3.24}$$

$$+2\sum_{k}^{N+1} \sqrt{\epsilon^{d=k}(h \circ g)}. \tag{3.25}$$

Extracting terms that belong to the test domain $d = N+1$ leads to

$$\sum_{l=1}^{N} \text{JSD}(p(y|d=l)||p(y|d=N+1)) \tag{3.26}$$

$$+ \sum_{1\leq i<j\leq N} \text{JSD}(p(y|d=i)||p(y|d=j)) \tag{3.27}$$

$$\leq \sum_{l=1}^{N} \text{JSD}(p(\mathbf{z}|d=l)||p(\mathbf{z}|d=N+1)) \tag{3.28}$$

$$+ \sum_{1\leq i<j\leq N} \text{JSD}(p(\mathbf{z}|d=i)||p(\mathbf{z}|d=j)) \tag{3.29}$$

$$+2\sqrt{\epsilon^{d=N+1}(h\circ g)} + 2\sum_{k}^{N}\sqrt{\epsilon^{d=k}(h\circ g)} \tag{3.30}$$

Assuming we find a perfect intermediate representation $\mathbf{z}$ for all N training domains and the test domain $d = N+1$ (assuming such an $\mathbf{z}$ exists) we are left with

$$\sum_{l=1}^{N} \text{JSD}(p(y|d=l)||p(y|d=N+1)) \tag{3.31}$$

$$+ \sum_{1\leq i<j\leq N} \text{JSD}(p(y|d=i)||p(y|d=j)) \tag{3.32}$$

$$\leq 2\sqrt{\epsilon^{d=N+1}(h\circ g)} + 2\sum_{k}^{N}\sqrt{\epsilon^{d=k}(h\circ g)} \tag{3.33}$$

As it was the case for domain adaptation, we see that the joint risk across all domains (training and test) is lower bounded by the pairwise divergence of the marginal label distribution of all domains. Given the existence of an unobserved confounder, as seen in Figure 3.1, the marginal label distribution is unlikely to match.

However, there exists a multitude of domain generalization methods that do not explicitly address the problem of hidden confounders [Balaji et al., 2018, Carlucci et al., 2019b,a, Ding and Fu, 2018, Ghifary et al., 2015, Ilse et al., 2020, Li et al., 2018a, Mancini et al., 2018, Motiian et al., 2017, Shankar et al., 2018, Tzeng et al., 2014, Wang et al., 2018]. However, the majority of these methods are evaluate on benchmark datasets, e.g., VLCS [Khosla et al., 2012] or PACS [Li et al., 2017], where the domain d and the target y are confounded. As shown in Equation 3.33, this can result in poor generalization performance. Nonetheless, we cannot rule out the possibility that some of these methods can implicitly deal with confounders, thus achieving good generalization performance.

To the best of our knowledge, there are currently very few methods that address the issue of spuriously correlated domains d and targets

y [Arjovsky et al., 2020, Heinze-Deml and Meinshausen, 2021, Li et al., 2018b, Krueger et al., 2020], where Li et al. [2018b] extends the idea of domain adversarial learning to enforce conditional domain invariance, i.e., $p(\mathbf{z}|y,d=i) = p(\mathbf{z}|y,d=j)$.

3.6.9 Data augmentation

We will briefly summarize how data augmentation is currently viewed in the computer vision community for an in-depth survey see Shorten and Khoshgoftaar [2019]. In computer vision, data augmentation is seen as an effective technique for improving performance on various tasks such as image classification, object detection, and image segmentation. In the image domain, data augmentation techniques can be roughly divided into two categories:

1. Augmenting the geometry of an image: Commonly used transformations are rotations, horizontal and vertical flips, scaling, cropping, occlusion, and elastic deformations.

2. Augmenting the color of an image: Random values are added or subtracted from the color channels of an image. Instead of applying this transformation directly in the RGB colorspace, other color spaces like CIELAB and HSL are commonly used [Tellez et al., 2019].

Data augmentation combines the transformation listed above that are randomly applied to all images during training.

3.6.10 Data augmentation in application-focused research areas

In the following, we summarize two examples of the successful application of data augmentation for domain generalization in medical imaging and robotics. We want to highlight that in both examples, the actual task and the domains are spuriously correlated.

Histopathology The high variability of the appearance of histopathology images is a major obstacle for the deployment of automatic image analysis systems. The variability of appearance results from a multitude of preparation steps that are applied to the specimen: cutting, fixating, staining, and scanning. Each step introduces its artifacts. This leads to different color distributions among histopathology laboratories. Tellez et al. [2019] perform a detailed comparison of commonly used data augmentation, see Appendix Figure 3.9. The augmentation techniques consist of random rotation and flipping, random color perturbation, and color normalization. These augmentation techniques are compared on a dataset composed of histopathology images from nine different laboratories. We argue that

a hidden confounder exists that spuriously correlates the staining and scanner artifacts (caused by the laboratories) and the abnormalities in the tissue (caused by the diseases). By augmenting the color of the histopathology images Tellez et al. [2019] can learn features that are invariant to the laboratories. Furthermore, Tellez et al. [2019] find that random color perturbation outperforms color normalization. We argue that random color perturbation simulates noise interventions, whereas color normalization tries to simulate interventions where the color of a histopathology image is set to a fixed value. As described in Section 3.2, this requires first to estimate the color distribution of the original histopathology image, which is a challenging problem. As a result, data augmentation in the form of random color perturbation is better suited to simulate interventional data.

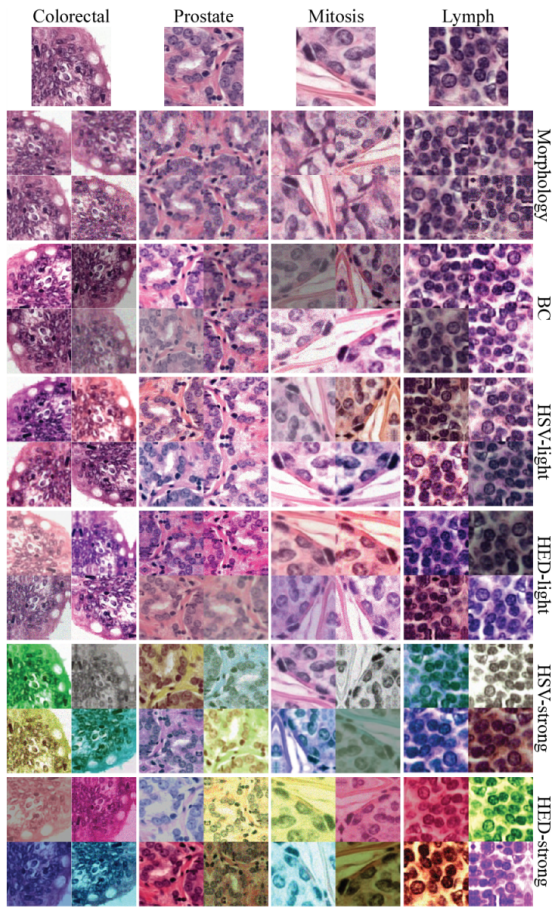


Figure 3.9. Domain randomization histopathology, taken from Tellez et al. [2019].

Robotics Performing robotic learning on physical hardware is often not feasible due to: (*i*) the large number of training samples that are required, and (*ii*) potential damage to the hardware if the learning relies on random exploration. Therefore, learning in a physics simulator is of great interest. While learning in a simulator is cheap and safe, we are facing a new problem, namely, how to overcome the so-called *reality gap*, i.e., the differences between simulation and the real world. In Tobin et al. [2017] they focus on a robotic manipulation task that involves a robotic arm and eight 3D objects that are placed on a table. In this scenario, a neural network is used to detect the location of an object. To be able to generalize from the simulation to the real world, Tobin et al. [2017] apply a variety of data augmentation techniques to the simulator, e.g., randomization of position and texture of all objects on the table, textures of the table, floor, skybox, and robot, and the addition of random noise. We argue that there exists a hidden confounder that introduces a spurious correlation between, e.g., the lighting conditions and the location of the objects on the table. By applying heavy data augmentation during the training process they are able to generalize to unseen lighting conditions in the real world.

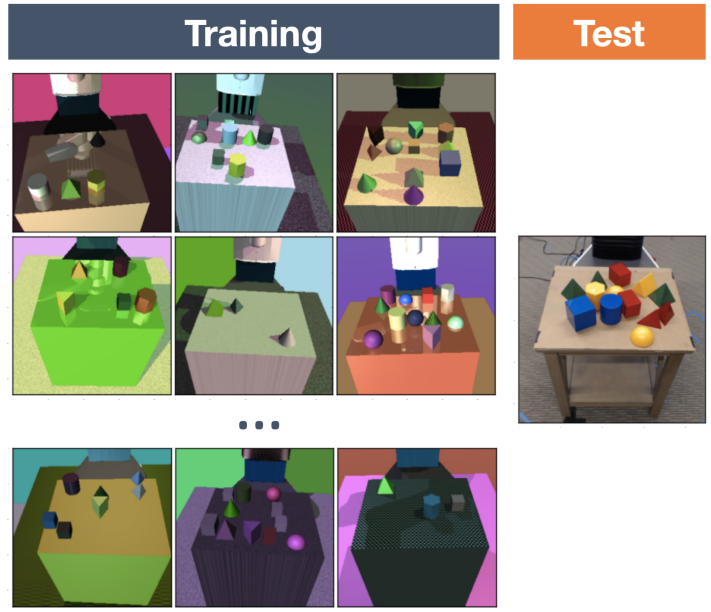


Figure 3.10. Domain randomization in robotics, taken from Tobin et al. [2017].

4. Efficient Causal Inference from Combined Observational and Interventional Data through Causal Reductions

We now consider the case where we have no apriori knowledge about the symmetries in the data. Therefore we cannot use the invariant architectures or data augmentation techniques presented in Section 1.2.1. As seen in Section 1.2.2, we need to make additional assumptions about the data generating process to train invariant machine learning models. Suppose we have access to experimental data, e.g., from an RCT. In that case, we can directly learn invariant machine learning models, in contrast to Chapter 3, where we simulated interventions using data augmentation. However, in most practical settings, we won't have access to many interventional samples. Therefore, training machine learning models purely from interventional samples can lead to high variance estimates. By adding observational samples, we can potentially reduce the variance of the estimator. However, we are at risk of introducing bias. In the following, we show how to use normalizing flows to learn interventional and observational distribution with a single model in this instance. We exploit the fact that parts of the parameters can be jointly trained using interventional and observational data. The parameter sharing allows us to reduce the variance without introducing bias.

4.1 Introduction

In this work, we propose a novel principled approach for causal effect estimation that can efficiently combine observational and interventional samples, even in the presence of unobserved confounding in the observational regime. We show that this method can potentially reduce the required Randomized Controlled Trial (RCT) sample size when sufficient observational samples are available (e.g., in the form of electronic health records). Recent real-world examples that could benefit from such an approach are the COVID-19 vaccine trials. Several of the vaccines require two dosages. For example, the interval during the vaccine trials was 21 days between doses for the Pfizer vaccine and 28 days for the Moderna

vaccine. However, due to a shortage of supplies and logistical challenges, the second dosage is delayed in many countries. The question then arises: What is the effect of the time between the first and the second dosage on the vaccine efficacy? In the absence of any large randomized controlled trials that provide a definite answer to this question, one may hope to estimate this by combining the few available clinical trial data with massive global observational data collected as a part of the different vaccination campaigns performed worldwide. The method we propose here provides a principled approach for such causal inference problems.

The main complication when estimating causal effects is the potential presence of observed and—in particular—unobserved *confounders* (common causes of the cause and the effect). Our key technical contribution, which we believe to be a valuable tool on its own, is a construction that typically *reduces* the size of the latent confounder space in a structural causal model (or causal Bayesian network with latent confounders). This construction only assumes the absence of causal feedback from outcome to treatment. The data was not subject to selection bias due to implicit conditioning on common effects of treatment and outcome.

This *causal reduction* operation shows that without loss of generality, one only needs to model a single latent confounding variable that lives in the same space as the treatment variable, even if, in reality, there could be many latent confounders and their joint space might be much larger. In particular, for a real-valued, one-dimensional treatment variable, a real-valued, one-dimensional confounder suffices. The causal reduction is a key step towards a parsimonious joint parameterization of the observational and interventional distributions.

For the linear-Gaussian case, we prove that our reduced parameterization implies that the observational and interventional distributions are not independent but are related by equality constraints. This complements existing work on inequality constraints in the case of discrete treatment and outcome variables (Bell [1964], Balke and Pearl [1997], Wolfe et al. [2019]). We conjecture that such dependencies between the observational and interventional distribution hold more generally (i.e., not only in the linear-Gaussian or discrete settings) and provide empirical support for this conjecture.

To make progress in the general nonlinear setting, we parameterize the reduced causal model using a flexible class of easily invertible nonlinear transformations, so-called normalizing flows [Tabak and Turner, 2013, Rezende and Mohamed, 2015]. Normalizing flows enables the use of a simple multi-task maximum-likelihood approach to estimate the reduced model parameters, where one can now combine observational and interventional training data, allowing for latent confounding in the observational regime.

We perform a series of experiments on data simulated using nonlinear causal mechanisms. We find that we can significantly reduce the number of interventional samples required to achieve a certain accuracy when adding sufficient observational training samples. We observe that parameter sharing allows one to learn a more accurate model from a combination of data than from each subset individually. This suggests that this approach successfully exploits the conjectured dependence between the observational and interventional distributions, and opens up practical applications and further theoretical questions regarding the precise nature of the relationship between observational and interventional distributions.

In summary, our three main contributions are: (i) A causal reduction method that replaces arbitrary latent confounders with a single latent confounder that lives in the same space as the treatment variable, without changing the observational and interventional distributions entailed by the causal model; (ii) A flexible parameterization of the reduced model using normalizing flows, which enables us to estimate the observational and interventional distributions by jointly learning from observational and interventional data without making strong parametric assumptions; (iii) A derivation of equality constraints between interventional and observational distributions entailed by linear Gaussian causal models.

4.2 Related work

Prior work on causal inference from multiple datasets can be roughly divided into addressing three different tasks: (i) identifying causal effects when the causal structure is known (see, e.g., Lee et al. [2020] and references therein), (ii) discovering/learning the causal structure (see, e.g., Mooij et al. [2020] and references therein), and (iii) estimating causal effects. The present work addresses the latter task, focussing on continuous treatment and outcome.

There exists a plethora of work on estimating causal effects solely from observational data. The vast majority of proposed methods assumes that a set of observed variables can be used to adjust for all confounding factors [Colnet et al., 2020]. Unfortunately, one can never test this assumption, and the reliability of the conclusions of such observational studies is debated [Madigan et al., 2014]. While most work considers the case in which treatment is binary, Hirano and Imbens [2005] generalize the propensity score for continuous treatment variables.

Researchers recently started looking into combining those different modalities to address the limitations of causal effect estimation from interventional or observational data alone. Most prior work on this topic still relies on the assumption that all confounders are observed in the observational regime (e.g., Silva [2016], Rosenman et al. [2018]). We only found three papers that allow for latent confounding in the observational regime.

The method by Rosenman et al. [2020] attempts to reduce confounding bias rather than completely removing it. For binary treatments, Kallus et al. [2018] rely on an additional assumption that the hidden confounder has a certain parametric structure that can be modeled effectively (which may also reduce confounding bias). In contrast, Athey et al. [2020] depend on observed short-term and long-term outcome variables. In contrast, our approach can completely remove confounding bias without making such additional assumptions.

Another approach that sidesteps the strong untestable assumption of no unobserved confounding is to *bound* the causal effect in terms of properties of observational data [Balke and Pearl, 1997, Pearl, 1995]. While these bounds are valid in the presence of arbitrary unobserved confounding, they are often too loose to be of practical relevance and only hold for discrete treatment variables. Recently, Wolfe et al. [2019] introduced a technique called inflation that can be used to derive tighter bounds.

Furthermore, methods that do not rely on bounds or an adjustment set have to make other untestable assumptions on the causal mechanism. For example, Angrist et al. [1996], Kilbertus et al. [2020], Gunsilius [2020] rely on the existence of instrumental variables that are not affected by unobserved confounders and on restrictions of the model space. Miao et al. [2018] and Louizos et al. [2017] assume proxy variables that, while being correlated with unobserved confounders, do not confound the treatment and outcome themselves. Last, the deconfounder of Wang and Blei [2019] builds on the assumptions that there are no unobserved single-cause confounders.

4.3 Theory

For simplicity of exposition, we will make some assumptions regarding the types of variables below, but the construction of the causal reduction can be done for any standard measurable spaces. Furthermore, we will only consider perfect (hard/atomic/surgical) interventions and make use of the corresponding do-*operator* notation introduced by Pearl [2009].

4.3.1 Reduction of the latent space

Consider a treatment variable $\mathbf{X} \in \mathcal{X} = \mathbb{R}^M$ and an outcome variable $\mathbf{Y} \in \mathcal{Y} = \mathbb{R}^N$. We assume that the outcome does not cause the treatment. Furthermore, let there exist K latent confounders $Z_1, \dots, Z_K$, where $Z_i \in \mathcal{Z}_i = \mathbb{R}$, with an arbitrary dependency structure, see Figure 4.1 (a) for the corresponding Directed Acyclic Graph (DAG). Without loss of generality, we can summarize the K latent confounders $Z_1, \dots, Z_K$ with arbitrary dependency

structure using a single latent confounder $\mathbf{Z} \in \mathcal{Z} = \mathbb{R}^K$:

$$p(\mathbf{x},\mathbf{y}) = \int_{\mathcal{Z}_1} \cdots \int_{\mathcal{Z}_K} p(\mathbf{x},\mathbf{y},z_1,\dots,z_K) dz_1 \dots dz_K = \int_{\mathcal{Z}} p(\mathbf{x},\mathbf{y},\mathbf{z}) d\mathbf{z}. \tag{4.1}$$

The resulting causal Bayesian network is shown in Figure 4.1 (b), which has the following factorization:

$$p(\mathbf{x},\mathbf{y},\mathbf{z}) = p(\mathbf{y}|\mathbf{x},\mathbf{z}) p(\mathbf{x}|\mathbf{z}) p(\mathbf{z}). \tag{4.2}$$

We aim to replace the above causal Bayesian network with one that is interventionally equivalent with respect to interventions on $\mathbf{X}$ and $\mathbf{Y}$, but where the latent confounder space $\mathcal{Z}$ is lower-dimensional.

First, we generate a copy $\mathbf{W} := \mathbf{X}$ of the treatment variable $\mathbf{X}$. We will interpret $\mathbf{W}$ as a latent variable and $\mathbf{X}$ as an observed deterministic effect of $\mathbf{W}$, via the function $\mathbf{X} = \mathrm{id}(\mathbf{W})$. We obtain the Bayesian Network in Figure 4.1 (c):

$$p(\mathbf{x},\mathbf{y},\mathbf{w},\mathbf{z}) = p(\mathbf{y}|\mathbf{x},\mathbf{z}) p(\mathbf{x}|\mathbf{w}) p(\mathbf{w}|\mathbf{z}) p(\mathbf{z}), \tag{4.3}$$

where $p(\mathbf{w}|\mathbf{z}) := p(\mathbf{x}|\mathbf{z})|_{\mathbf{x}=\mathbf{w}}$ is a copy of the Markov kernel from above evaluated in $\mathbf{w}$ rather than in $\mathbf{x}$. Furthermore, $p(\mathbf{x}|\mathbf{w}) := \delta_{\mathbf{w}}(\mathbf{x})$ is the delta peak centered at $\mathbf{w}$, representing the deterministic identity map from $\mathbf{W}$ to $\mathbf{X}$. If we marginalize out $\mathbf{W}$ we arrive at the initial causal Bayesian network in Figure 4.1 (b) again. Since interventions on observed variables commute with marginalizing over latent variables [Bongers et al., 2020], the Bayesian networks in Figure 4.1 (b) and (c) are interventionally equivalent with respect to interventions on $\mathbf{X}$ and $\mathbf{Y}$.

Second, we refactorize the latent distribution as shown in Figure 4.1 (c), (d) and (e):

$$p(\mathbf{x},\mathbf{y},\mathbf{w},\mathbf{z}) = p(\mathbf{y}|\mathbf{x},\mathbf{z}) p(\mathbf{x}|\mathbf{w}) p(\mathbf{w}|\mathbf{z}) p(\mathbf{z}) \tag{4.4}$$

$$= p(\mathbf{y}|\mathbf{x},\mathbf{z}) p(\mathbf{x}|\mathbf{w}) p(\mathbf{w},\mathbf{z}) \tag{4.5}$$

$$= p(\mathbf{y}|\mathbf{x},\mathbf{z}) p(\mathbf{x}|\mathbf{w}) p(\mathbf{z}|\mathbf{w}) p(\mathbf{w}). \tag{4.6}$$

The Bayesian networks representing these three factorizations are interventionally equivalent with respect to interventions on $\mathbf{X}$ and $\mathbf{Y}$, as we only factor the latent distributions differently and do not consider interventions on the latent variables.

Last, we can marginalize over $\mathbf{Z}$ and obtain:

$$p(\mathbf{x},\mathbf{y},\mathbf{w}) = p(\mathbf{y}|\mathbf{x},\mathbf{w}) p(\mathbf{x}|\mathbf{w}) p(\mathbf{w}), \tag{4.7}$$

where we used the following composed Markov kernel:

$$p(\mathbf{y}|\mathbf{x},\mathbf{w}) := \int p(\mathbf{y}|\mathbf{x},\mathbf{z}) p(\mathbf{z}|\mathbf{w}) d\mathbf{z}. \tag{4.8}$$

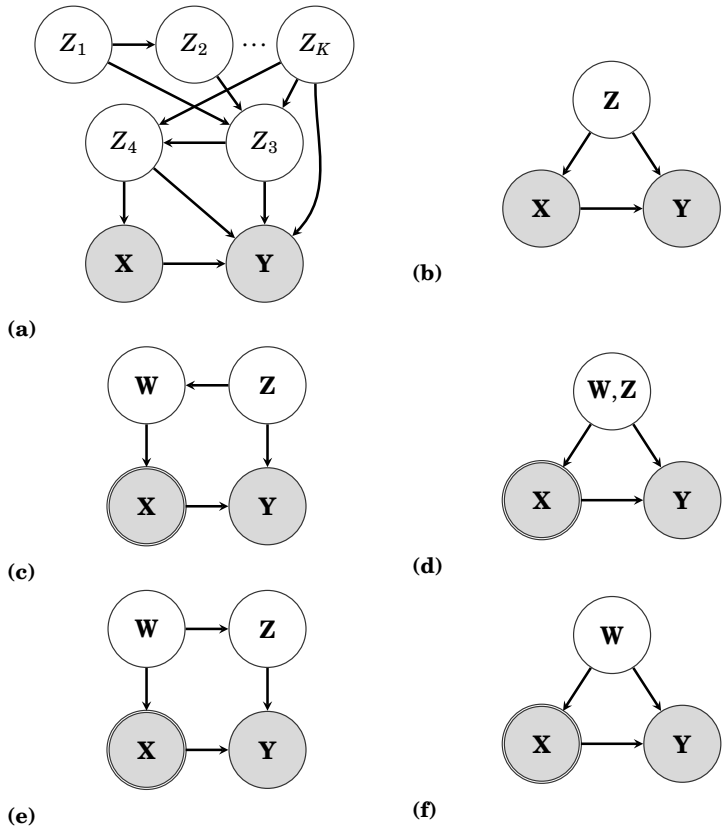


Figure 4.1. A graphical explanation of our causal reduction technique. (a) We assume a treatment variable $\mathbf{X}$, an outcome variable $\mathbf{Y}$, and K latent confounders $Z_1,\dots,Z_K$ with an arbitrary dependency structure. (b) We represent the K latent confounders $Z_1,\dots,Z_K$ by $\mathbf{Z} \in \mathcal{Z} = \mathbb{R}^K$. (c) We create a copy of $\mathbf{X}$ called $\mathbf{W}$. We use a double circle to indicate that a variable is a deterministic function of its parents. (d, e) Instead of using the factorization from (c), $p(\mathbf{w},\mathbf{z}) = p(\mathbf{w}|\mathbf{z})p(\mathbf{z})$, we choose $p(\mathbf{w},\mathbf{z}) = p(\mathbf{z}|\mathbf{w})p(\mathbf{w})$. (f) Last, we marginalize over $\mathbf{Z}$. Note that at every step (a–f) the Bayesian networks are interventionally equivalent with respect to interventions on $\mathbf{X}$ and $\mathbf{Y}$.

Again, since marginalizing over latent variables and interventions commute, the final Bayesian network in Figure 4.1 (f) is interventionally equivalent to the ones in Figure 4.1 (a–e) with respect to interventions on $\mathbf{X}$ and $\mathbf{Y}$.

Since $\mathbf{W}$ is a copy of $\mathbf{X}$, we successfully reduced the dimensionality of the latent confounder from K to M (assuming $M < K$). In the common case of one-dimensional $\mathbf{X}$, we expect $M = 1 \ll K$ and therefore achieve a significant reduction of the latent space. We formulate the conclusion as a theorem:

Theorem 4.3.1 (Causal Reduction)**.** *Let $\mathcal{M}$ be a causal Bayesian network with observed variables $\mathbf{X} \in \mathcal{X}, \mathbf{Y} \in \mathcal{Y}$ and latent variables, $Z_1 \in \mathcal{Z}_1, \dots, Z_K \in \mathcal{Z}_K$ such that $\mathbf{Y}$ is not an ancestor of $\mathbf{X}$.*

Then there exists a causal Bayesian network $\mathcal{M}^*$ with observed variables $\mathbf{X} \in \mathcal{X}$ and $\mathbf{Y} \in \mathcal{Y}$ and a single latent confounder $\mathbf{W} \in \mathcal{X}$ (that takes values in the same space as $\mathbf{X}$) such that $\mathcal{M}^*$ is interventionally equivalent to $\mathcal{M}$ with respect to interventions on the observed variables $\mathbf{X}$ and $\mathbf{Y}$:

$$p_{\mathcal{M}}(\mathbf{x},\mathbf{y}) = p_{\mathcal{M}^*}(\mathbf{x},\mathbf{y})$$

$$p_{\mathcal{M}}(\mathbf{x} \mid \mathrm{do}(\mathbf{y})) = p_{\mathcal{M}^*}(\mathbf{x} \mid \mathrm{do}(\mathbf{y}))$$

$$p_{\mathcal{M}}(\mathbf{y} \mid \mathrm{do}(\mathbf{x})) = p_{\mathcal{M}^*}(\mathbf{y} \mid \mathrm{do}(\mathbf{x}))$$

We call the causal Bayesian network $\mathcal{M}^*$ a *causal reduction* of $\mathcal{M}$ since it will typically be the case that the latent space will be reduced, yet the causal semantics are preserved by construction. The single latent confounder $\mathbf{Z}$ in $\mathcal{M}^*$ will parsimoniously represent the causal influence of *all* latent confounders of $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{M}$. For example, a single binary confounder suffices for a binary treatment variable. Extending the derivation to *simple* structural causal models (an extension of causal Bayesian networks that can represent feedback loops [Bongers et al., 2020]) is straightforward, as long as $\mathbf{X}$ and $\mathbf{Y}$ are not part of a causal cycle (although the other variables might be involved in cycles).

4.3.2 From causal Bayesian networks to structural causal models

Whereas in the previous section, we relied on causal Bayesian networks to conduct our reduction, we now move to Structural Causal Models (SCMs) [Pearl, 2009, Bongers et al., 2020] to obtain convenient parameterizations. We make use of the exogenous variables $\mathbf{U},\mathbf{V}$ to represent the noise in the reduced causal model. This, in turn, allows us to express all causal relationships as deterministic functions. Estimating the model then boils down to estimating these functions, as we will illustrate in Section 4.4.

Theorem 4.3.2. *Let $P(\mathbf{X}|\mathbf{Y})$ be a Markov kernel (e.g. a conditional probability distribution) of a $\mathbb{R}^M$-valued variable $\mathbf{X}$ with components X_m, $m = 1,\dots,M$*

and with argument $\mathbf{Y}$ *that can take values in any measurable space. Then there exists a* M*-dimensional standard normal random variable* $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_M)$ *independent of* $\mathbf{Y}$ *and a deterministic measurable map* F *such that:*

$$\mathbf{X} = F(\mathbf{Z}, \mathbf{Y}) \quad a.s. \tag{4.9}$$

Furthermore, the map F *is 'well-behaved', in the sense that it is composed out of (inverse) conditional cumulative distribution functions.*

The proof is provided in the Appendix 4.7.2. Theorem 4.3.2 enables us to obtain a reduced SCM from the reduced causal Bayesian network in Equation 4.7 with structural equations

$$\mathbf{X} = F(\mathbf{U}), \tag{4.10}$$

$$\mathbf{Y} = G(\mathbf{U}, \mathbf{V}, \mathbf{X}), \tag{4.11}$$

where $\mathbf{U} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_M) \perp\!\!\!\perp \mathbf{V} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}_N\right)$, and F, G are deterministic maps. This SCM encodes the same observational distribution $p(\mathbf{x}, \mathbf{y})$ and interventional distributions $p(\mathbf{y}|\mathrm{do}(\mathbf{x})), p(\mathbf{x}|\mathrm{do}(\mathbf{y}))$ as the causal Bayesian network. This allows us to parameterize the reduced causal model in terms of the two functions F and G.

4.3.3 Parameter sharing in the linear Gaussian case

We now consider the case where all causal relationships in Figure 4.1 (a) are linear, and all distributions are Gaussian. We can then guarantee that the reduced causal model is linear Gaussian as well. The precise statement and proof are delegated to Appendix 4.7.3.

We use the reduced linear Gaussian model from Corollary 4.7.2 to prove that the parameters of the interventional distribution constrain the parameters of the observational distribution.

Theorem 4.3.3 (Linear Gaussian parameter constraints)**.** *Consider a linear-Gaussian SCM (or causal Bayesian network with possible latent variables) with two observed variables* $\mathbf{X}$ *and* $\mathbf{Y}$ *such that* $\mathbf{Y}$ *is not ancestor of* $\mathbf{X}$*. The entailed observational and interventional distributions are Gaussian. Modeling* $p(\mathbf{x}), p(\mathbf{y}|\mathbf{x})$ *and* $p(\mathbf{y}|\mathrm{do}(\mathbf{x}))$ *independently from each other could be done with the following parameterization:*

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\alpha}, \Sigma), \tag{4.12}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}\left(\mathbf{y}|\boldsymbol{\gamma} + \Delta\mathbf{x}, \Pi\right), \tag{4.13}$$

$$p(\mathbf{y}|\mathrm{do}(\mathbf{x})) = \mathcal{N}\left(\mathbf{y}|\widetilde{\boldsymbol{\gamma}} + \widetilde{\Delta}\mathbf{x}, \widetilde{\Pi}\right), \tag{4.14}$$

with covariance matrices Σ*,* Π*,* $\widetilde{\Pi}$*. However, using the reduced causal model from Corollary 4.7.2 we find that these parameters are constrained by the*

following relations:

$$\left(\widetilde{\boldsymbol{\gamma}} - \boldsymbol{\gamma}\right) + \left(\widetilde{\Delta} - \Delta\right)\boldsymbol{\alpha} = 0, \tag{4.15}$$

$$\left(\widetilde{\Delta} - \Delta\right)\Sigma\left(\widetilde{\Delta} - \Delta\right)^{\top} + \Pi = \widetilde{\Pi}. \tag{4.16}$$

From Equation 4.16 we can easily see that $\widetilde{\Pi} - \Pi$ is positive semidefinite. Furthermore, we see that these constraints lead to a reduced parameter count, N parameters for Equation 4.15 and $N(N+1)/2$ parameters for Equation 4.16, assuming $\mathbf{y}$ to be N-dimensional. In total, we have reduced the parameter count by $N(N+3)/2$ by modeling the parameters of the observational and interventional distributions jointly. The proof of Theorem 4.3.3 can be found in Appendix 4.7.4.

Now consider the task of learning the parameters of our reduced model. In the linear Gaussian case, the reduced causal model tells us exactly how many parameters we need to model the observational and interventional distribution and which parameters are shared. Since the parameters $\mathbf{c}, D, E$ and, F are shared between the observational and interventional distribution. We can estimate them jointly using observational and interventional data, effectively reducing sample complexity when trying to model the interventional distribution. This is beneficial for causal effect estimation when we assume that we only have access to a small number of interventional samples and a large number of observational samples. In the Appendix 4.7.6, we experiment on observational and interventional data generated with linear causal mechanisms, giving a linear parameterization of the reduced linear model that can learn linear causal mechanisms.

4.3.4 Reduction with observed confounders

There are many scenarios where we are interested in estimating the conditional causal effect of interventions given additional covariates $\mathbf{C}$ that might confound treatment and outcome, for example when estimating the efficacy of a vaccine depending on age, weight or gender. We consider an additional set of L *observed* confounders $C_1, \ldots, C_L$ of $\mathbf{X}$ and $\mathbf{Y}$, taking values in arbitrary measurable spaces, and with an arbitrary joint distribution $p(\mathbf{c})$. In the following, we summarize all L observed confounders using a single variable $\mathbf{C} \in \mathscr{C}$. We provide a more detailed derivation in the Appendix 4.7.5 and give only a short sketch here. First, we follow the same steps as in Section 4.3.1 to derive a reduced causal model of the following form

$$p(\mathbf{x}, \mathbf{y}, \mathbf{w}, \mathbf{c}) = p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \mathbf{c})p(\mathbf{x}|\mathbf{w})p(\mathbf{w}|\mathbf{c})p(\mathbf{c}). \tag{4.17}$$

Again, at every step, the Bayesian network is interventionally equivalent to the ones before w.r.t. interventions on $\mathbf{X}$ and $\mathbf{Y}$. Then, we use a similar approach as in Section 4.3.2 but also marginalize out $\mathbf{W}$ to convert the

causal Bayesian Network into an SCM with structural equations of the form

$$\mathbf{X} = F(\mathbf{U}, \mathbf{C}), \tag{4.18}$$

$$\mathbf{Y} = G(\mathbf{U}, \mathbf{V}, \mathbf{X}, \mathbf{C}), \tag{4.19}$$

where $\mathbf{U} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_M) \perp\!\!\!\perp \mathbf{V} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}_N\right)$ and F and G are two deterministic maps. Again, this preserves the observational and interventional distributions w.r.t. interventions on $\mathbf{X}$ and $\mathbf{Y}$.

4.4 Practical implementation

Now that we have successfully reduced the model complexity, we will parameterize the functions F and G to learn the model from data. Intuitively, the reduced SCM in Section 4.3.2 tells us that the parameters of G are shared among observational and interventional samples for an intervention on $\mathbf{X}$, whereas the parameters of F are not. While this can be done in many different ways, we use diffeomorphisms, i.e., mappings that are differentiable and have a differentiable inverse. Using the change-of-variables formula, we can derive a maximum-likelihood estimator for the mappings' parameters that can be efficiently optimized through backpropagation. In the deep learning community, those invertible and differentiable mappings are called normalizing flows, and much recent research went into finding flexible and easily invertible mappings, see Pawlowski et al. [2020] and Khemakhem et al. [2020] for other recent applications of normalizing flows to approximate nonlinear causal mechanisms. Our flow model consists of two flows where the first flow is trained using observational data and the second flow is trained using observational and interventional data.

In the following, we derive the loss function for observational and interventional data separately. For the remaining part of this paper we focus on one-dimensional treatment outcome pairs, i.e. $x \in \mathcal{X} = \mathbb{R}$ and $y \in \mathcal{Y} = \mathbb{R}$, and (optionally) a L-dimensional observed confounder $\mathbf{c} \in \mathcal{C} = \mathbb{R}^L$.

4.4.1 Observational data

Following the SCM in Equation 4.11 the joint-likelihood $p(x, y)$ can be factorized as $\log p(x, y) = \log p(y|x) + \log p(x)$. We now use the following bijective transformations between observed variables x, y and latent variables u, v

$$u = f_{\boldsymbol{\phi}}(x), \tag{4.20}$$

$$v = g_{x,u;\boldsymbol{\theta}}(y), \tag{4.21}$$

where the function $g(.)$ is invertible with respect to v. Here, $f_{\boldsymbol{\phi}} = F^{-1}$ from Equation 4.10 and $g_{x,u;\boldsymbol{\theta}}$ is the inverse of $v \mapsto G(u, v, x)$ from Equation 4.11

(for fixed $u, x, \boldsymbol{\theta}$). Without loss of generality we assume independent, standard Gaussian distributions for u, v: $p_U(u) = \mathcal{N}(0,1) \perp\!\!\!\perp p_V(v) = \mathcal{N}(0,1)$. The transformations defined above allow us to rewrite the joint likelihood using the change of variable formula

$$\begin{aligned}\log p(x, y) &= \log p_V(g_{x,u;\boldsymbol{\theta}}(y)) + \log \left| \frac{\partial g_{x,u;\boldsymbol{\theta}}(y)}{\partial y} \right| + \log p_U(f_{\boldsymbol{\phi}}(x)) + \log \left| \frac{\partial f_{\boldsymbol{\phi}}(x)}{\partial x} \right|. \\ &= \log p_V(g_{x,f_{\boldsymbol{\phi}}(x);\boldsymbol{\theta}}(y)) + \log \left| \frac{\partial g_{x,f_{\boldsymbol{\phi}}(x);\boldsymbol{\theta}}(y)}{\partial y} \right| + \log p_U(f_{\boldsymbol{\phi}}(x)) + \log \left| \frac{\partial f_{\boldsymbol{\phi}}(x)}{\partial x} \right|.\end{aligned} \tag{4.22}$$

where in the last step, we substituted $u = f_{\boldsymbol{\phi}}(x)$ into $g_{x,u;\boldsymbol{\theta}}(y)$. The parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ are jointly updated by minimizing $\sum_{o=1}^{N_O} -\log p(x_o, y_o)$ given N_O observational training samples.

4.4.2 Interventional data

In contrast to the observational setting, we only have to consider the conditional likelihood $p(y|\mathrm{do}(x))$ in the interventional case. Since we cannot use $f_{\boldsymbol{\phi}}(x)$ to impute u, we instead marginalize over u

$$\log p(y \mid \mathrm{do}(x)) = \log \int p(y|\mathrm{do}(x), u) p(u) du. \tag{4.23}$$

Inserting the bijective mapping $v = g_{x,u;\boldsymbol{\theta}}(y)$ in Equation 4.23, we obtain

$$\log p(y \mid \mathrm{do}(x)) = \log \int p_V(g_{x,u;\boldsymbol{\theta}}(y)) \left| \frac{\partial g_{x,u;\boldsymbol{\theta}}(y)}{\partial y} \right| p(u) du, \tag{4.24}$$

where we use the trapezoidal rule to compute a numerical approximation of the integral. The parameter $\boldsymbol{\theta}$ can be updated by minimizing $\sum_{i=1}^{N_I} -\log p(y_i|\mathrm{do}(x_i))$ given N_I interventional training samples.

4.4.3 Joint optimization and sampling

Assuming we have N_O observational samples and N_I interventional samples, the full loss is given by

$$loss = \frac{1}{N_O} \sum_{o=1}^{N_O} -\log p(x_o, y_o) + \frac{1}{N_I} \sum_{i=1}^{N_I} -\log p(y_i|\mathrm{do}(x_i)). \tag{4.25}$$

The parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ of the transformation f and g are learned by minimizing the loss using gradient descent.

After training we are able to generate observational and interventional samples from $p(y \mid x)$ with a single flow model. The sampling procedure for observational samples consists of the following steps: $v \sim \mathcal{N}(0,1), u = f_{\boldsymbol{\phi}}(x_o)$, and $y_o = g^{-1}_{x_o,u;\boldsymbol{\theta}}(v)$, where we assume $x_o \in \mathbb{R}$ to be observed. If we instead want to generate an interventional sample from $p(y \mid \mathrm{do}(x))$, the

sample procedure follows: $v \sim \mathcal{N}(0,1), u \sim \mathcal{N}(0,1)$, and $y_i = g^{-1}_{x_i,u;\boldsymbol{\theta}}(v)$, where we assume $x_i \in \mathbb{R}$ to be observed. In order to parameterize the SCM in Equation 4.17 we simply have to replace the functions $f(.)$ and $g(.)$ by $u = f_{\mathbf{c};\boldsymbol{\phi}}(x)$ and $v = g_{x,u,\mathbf{c};\boldsymbol{\theta}}(y)$ where $\mathbf{c} \in \mathbb{R}^L$ is assumed to be observed. The optimization procedure does not change.

4.5 Experiments

Following the analysis in Section 4.3.3, we perform a series of experiments on simulated data, where the causal relationships between all variables are nonlinear, showing that we can significantly reduce the number of interventional samples required to estimate the interventional distribution $p(y|\text{do}(x))$ by training jointly with (possibly confounded) observational and interventional samples. Throughout this section, we are using the parameterization described in Section 4.4, where we use linear rational spline flows [Dolatabadi et al., 2020]. For a detailed description of this choice, see the Appendix 4.7.7. We perform two sets of experiments: (1) We consider K latent confounders $Z_1,\ldots,Z_K \in \mathbb{R}$ with an arbitrary dependency structure. (2) We consider L additional, observed confounders $C_1,\ldots,C_L \in \mathbb{R}$ with an arbitrary dependency structure. All flow models are implemented with the automatic differentiation packages Pytorch [Paszke et al., 2019] and Pyro [Bingham et al., 2019]. All code is available under `https://github.com/max-ilse/CausalReduction`.

4.5.1 Without observed confounders

Table 4.1. Comparison of a flow model trained with interventional samples only and a flow model trained with interventional and observational samples. We calculate the ratio N_I^*/N_I, where N_I^* is the number of interventional samples necessary to match the interventional test log-likelihood of a flow model trained with N_I interventional and 1000 observational samples. E.g. in the case of dataset 3 and $N_I = 100$, if we were to use only interventional samples, we would require twice as many interventional samples compared to using 100 interventional and 1000 observational samples. For dataset 11 to 15, we simulate an additional observed confounder $\mathbf{C}$. Note that if a large number of interventional samples ($250 < N_I \leq 1000$) are available the improvements become smaller as shown in the Appendix 4.7.10.

N_I	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
50	1.4	1.8	2.2	1.2	0.2	2.2	2.1	1.7	1.9	1.6	3.2	3.2	2.2	2.7	3.2
100	0.8	2.6	2.0	1.5	0.3	2.1	2.0	2.5	2.0	2.1	3.2	2.9	2.5	3.0	2.5
250	1.0	1.5	1.8	1.6	0.5	1.7	1.1	1.5	1.2	1.7	2.4	2.3	2.3	2.1	1.7

We simulate cause and effect pairs from the SCM with structural equations: $X = F(E_X,\mathbf{Z})$, $Y = G(X,E_Y,\mathbf{Z})$. A single dataset consists of observational and interventional samples. All causal relationships are simulated using fully connected neural networks with a single hidden layer, where

the weights are randomly initialized. The activation functions are REctified Linear Units (ReLUs). As a result, the simulated causal mechanisms are nonlinear. The values of $E_X, E_Y, \mathbf{Z}$ and $\text{do}(X)$ are sampled from a random distribution, as seen in Mooij et al. [2016]. A detailed step-by-step description of the simulation procedure is given in the Appendix 4.7.8. Following the process described above, we simulate 100 datasets while varying the number of dimensions K of the unobserved confounder $\mathbf{Z}$ and the random seed that is, among others, controlling the initialization of the neural networks used to model the causal mechanisms. We choose K between 1 and 10 since for $K > 10$, the joint distribution $p(x,y)$ becomes increasingly Gaussian due to the central limit theorem. Next, we manually select ten datasets with the smallest overlap of observational and interventional samples to select cases with "strong" confounding. Note that we choose these ten datasets before training a single flow model. A scatter plot of 1000 observational and 1000 interventional samples for each of the ten datasets can be found in Appendix 4.7.10.

In this experiment we are interested in estimating the interventional distribution $p(y|\text{do}(x))$. For each dataset, we train three variants of our reduced causal model parameterized with normalizing flows. The first flow model is trained using only observational data, see Section 4.4.1. The second flow model is trained using only interventional data, see Section 4.4.2. The third flow model is trained using observational and interventional data jointly, see Section 4.4.3. For each of the ten datasets, we keep the number of observational samples constant at 1000 and use an increasing number of interventional samples 50, 100, 250, 500, 750, 1000, resulting in six experiments per dataset. For example, in the case of 50 interventional and 1000 observational samples, the first flow model is trained with 1000 observational samples, the second flow model is trained with 50 interventional samples, and the third flow model is jointly trained with 1000 observational **and** 50 interventional samples. Motivated by the work of Oliver et al. [2018] on the realistic evaluation of semi-supervised learning algorithms, we use the same number of samples for training and validation. In every case, we use 1000 interventional samples for testing. To compare the performance of the three flow models, we calculate the negative log-likelihood averaged over the test set, $-\frac{1}{N_I}\sum_{i=1}^{N_I} \log p(y_i \mid \text{do}(x_i))$. To have a fair comparison, the same training procedure, architecture, optimizer, and hyperparameters are used for all flow models in all experiments. We use Adam [Kingma and Ba, 2015] with a learning rate of 0.001 and the default values for β_1, β_2. We train for 10000 epochs. The training is terminated early when the validation loss did not improve for 1000 epochs. We perform full batch gradient descent, where we alternate between batches of observational and interventional samples for the third flow model. For the linear rational spline flows, we use 32 bins and set the bound $B = 6$. We use a fully connected neural network with two hidden layers and ReLU

activations for the conditional version of the linear rational spline flows. In the Appendix 4.7.10, we provide extensive visualizations of the results of all experiments, including scatter plots of training data, samples from the trained flow models, negative log-likelihood values for all flow models on the interventional and observational test sets. To summarize our findings, we calculate the ratio of samples required to reach the same performance, measured in averaged negative log-likelihood when only using interventional samples. In Table 4.1 we see that in the case of dataset 3 and $N_I = 100$, we need two times the number of interventional samples (in the absence of observational training samples) to achieve the same performance as a flow model that is jointly trained with 100 interventional and 1000 observational samples. We can substantially reduce the number of interventional samples required when using an additional 1000 observational samples in eight of the ten datasets. Only in the case of dataset 5, we find that we need substantially more interventional samples to train our flow model jointly with observational and interventional data. We argue that in dataset 5, the interventional distribution resembles a standard Gaussian distribution that can easily be estimated from very few interventional samples. Last, the results in Table 4.1, dataset 1 to 10, are in agreement with qualitative results in the Appendix 4.7.10. We find that samples from the flow model trained with interventional and observational data better resemble the training data compared to samples from a flow model trained with only interventional data.

4.5.2 With observed confounders

We now consider the case of an additional L-dimensional observed confounder $\mathbf{C}$. We use the same setup as in Section 4.5.1 to simulate triples $(x, y, \mathbf{c})$. We use the following nonlinear causal mechanisms to generate treatment X and outcome Y: $X = f(E_X, \mathbf{Z}, \mathbf{C})$ and $Y = g(X, E_Y, \mathbf{Z}, \mathbf{C})$, a detailed description of the simulation procedure is given in the Appendix 4.7.8. Again, we generate 100 datasets by varying K, L between 1 and 5 and the random seed. We select five datasets following the same criteria as described in Section 4.5.1. Furthermore, we use the implementation described in Section 4.4.3 to estimate the SCM in Figure 4.3.4. For each of the five datasets, we keep the number of observational samples constant at 1000 and use an increasing number of interventional samples: 50, 100, 250, 500, 750, 1000, resulting in six experiments per dataset. We compare three flow models trained with observational, interventional, and observational plus interventional data, respectively. The training details are the same as in Section 4.5.1. An extensive comparison of the three flow models, as well as visualizations for each dataset, can be found in the Appendix 4.7.12. The main result of the experiments with additional observed confounders is the following: For each of the five datasets, we can substantially reduce

the required number of interventional samples with our flow model trained with observational and interventional data, see Table 4.1, dataset 11 to 15. We find that we can reduce the number of required samples by a factor of two to three when training with 1000 additional observational samples.

4.6 Conclusion

We propose a causal reduction technique that replaces any number of (possibly high-dimensional) unobserved confounders with a single confounder of the same dimensionality as the treatment variable, preserving the observational distributions entailed by the model and the interventional distributions for interventions on the treatment and outcome variable. Using a reduced model, we derive constraints between the observational and interventional distributions in the linear Gaussian case, showing that these objects are not independent. In the nonlinear case, we propose a flexible parameterization of the reduced causal model using normalizing flows. This parameterization allows us to train a single flow model by combining observational and interventional data. In simulations, for 13 out of 15 simulated datasets, we substantially reduce the required number of interventional samples if sufficient observational samples are available. Possible future work includes (i) applying the flow model to high dimensional outcome variables, e.g., medical images, (ii) using the reduction technique for causal discovery, e.g., inferring causal directions, and (iii) analyzing the relationship between the constraints in Section 4.3.3 and the instrumental and Bell inequalities [Pearl, 1995, Bell, 1964].

4.7 Appendix

4.7.1 Theorem A.1 and proof

Theorem 4.7.1. *Let $P(X|\mathbf{Y})$ be a Markov kernel, where the variable X takes values in $\mathbb{R}$ (or $[-\infty, \infty]$) and argument $\mathbf{Y}$ has values in any measurable space (e.g. $\mathbb{R}^N$). Then there exists a uniformly distributed variable $E \sim U[0,1]$ that is independent of $\mathbf{Y}$ and a deterministic function F, namely the conditional quantile function of X given $\mathbf{Y}$, such that:*

$$X = F(E|\mathbf{Y}) \quad a.s. \tag{4.26}$$

Proof. Consider the interpolated conditional cumulative distribution function (iccdf) of X given $\mathbf{Y}$ with $u \in [0,1]$:

$$G(x; u|\mathbf{y}) := P(X < x|\mathbf{y}) + u \cdot P(X = x|\mathbf{y}). \tag{4.27}$$

Furthermore, consider the conditional quantile function (cqf) of X given $\mathbf{Y}$ with $e \in [0,1]$:

$$F(e|\mathbf{y}) := \inf\{\tilde{x} \in \mathbb{R} \,|\, G(\tilde{x};1|\mathbf{y}) \geq e\}. \tag{4.28}$$

Then take any uniformly distributed random variable $U \sim U[0,1]$ independent of $(X,\mathbf{Y})$ and define:

$$E := G(X;U|\mathbf{Y}), \tag{4.29}$$

where we plugged X, U and $\mathbf{Y}$ into G. Then one can check using standard arguments for cdf and cqf that E is uniformly distributed, $E \sim U[0,1]$, which is independent of the value $\mathbf{y}$ of $\mathbf{Y}$. Furthermore, one can show that:

$$X = F(E|\mathbf{Y}) \quad \text{a.s.} \tag{4.30}$$

A detailed proof can be found in Forré [2021] in Appendix G. □

4.7.2 Proof of Theorem 3.2

Proof. We use Theorem 4.7.1 inductively.

1. Consider the cqf F_1 of $P(X_1|\mathbf{Y})$. Then by 4.7.1 there is a random variable $E_1 \sim U[0,1]$ independent of $\mathbf{Y}$ such that $X_1 = F_1(E_1|\mathbf{Y})$ a.s.

2. Now consider the cqf F_2 of $P(X_2|E_1,\mathbf{Y})$. Then by 4.7.1 there is a random variable $E_2 \sim U[0,1]$ independent of E_1, $\mathbf{Y}$ such that $X_2 = F_2(E_2|E_1,\mathbf{Y})$ a.s.

3. Now consider the cqf F_3 of $P(X_3|E_2,E_1,\mathbf{Y})$. Then by 4.7.1 there is a random variable $E_3 \sim U[0,1]$ independent of E_2,E_1, $\mathbf{Y}$ such that $X_3 = F_3(E_3|E_2,E_1,\mathbf{Y})$ a.s.

4. and so on .... until:

5. $X_M = F_M(E_M|E_{M-1},\ldots,E_1,\mathbf{Y})$ a.s. with $E_M \sim U[0,1]$ independent of $E_{M-1},\ldots,E_1$, $\mathbf{Y}$.

Now we put $Z_d := \Phi^{-1}(E_d)$, where Φ is the cdf of $\mathcal{N}(0,1)$. Then $E_d = \Phi(Z_d)$ and the Z_d are $\mathcal{N}(0,1)$-distributed and $\mathbf{Z} = (Z_1,\ldots,Z_M)$ is independent $\mathbf{Y}$). So $\mathbf{Z} = (Z_1,\ldots,Z_M) \sim \mathcal{N}(\mathbf{0},\mathbf{I}_M)$ and independent of $\mathbf{Y}$. Furthermore, we have almost surely the equations:

$$X_1 = F_1(\Phi(Z_1)|\mathbf{Y}), \tag{4.31}$$

$$X_2 = F_2(\Phi(Z_2)|\Phi(Z_1),\mathbf{Y}), \tag{4.32}$$

$$\vdots \qquad \ddots \tag{4.33}$$

$$X_M = F_M(\Phi(Z_M)|\Phi_{(Z_{M-1})},\ldots,\Phi(Z_1),\mathbf{Y}). \tag{4.34}$$

□

4.7.3 Corollary of Theorem 3.1: the linear Gaussian case

Here we state and prove a special case of the causal reduction for the linear Gaussian case.

Corollary 4.7.2 (Reduced linear Gaussian model)**.** *Consider a linear Gaussian SCM (or causal Bayesian network with possible latent variables) with observed variables* $\mathbf{X}$ *and* $\mathbf{Y}$ *such that* $\mathbf{Y}$ *is not ancestor of* $\mathbf{X}$*. Then this causal model is interventionally equivalent to a reduced linear Gaussian causal model with the following structural equations:*

$$\mathbf{X} = \mathbf{a} + B\mathbf{U}, \tag{4.35}$$

$$\mathbf{Y} = \mathbf{c} + D\mathbf{X} + E\mathbf{U} + F\mathbf{V}, \tag{4.36}$$

with vectors $\mathbf{a}$*,* $\mathbf{c}$ *and matrices B, D, E, F, where B and F can be chosen to be lower-triangular with non-negative diagonal entries, and where* $\mathbf{U}$ *is a standard Gaussian latent variable of the same dimension as* $\mathbf{X}$ *and where* $\mathbf{V}$ *is a standard Gaussian latent variable of the same dimension as* $\mathbf{Y}$ *that is independent of* $\mathbf{U}$*.*

Proof. This follows the same steps as the general construction in Equations 4.2, 4.3, 4.4, 4.5, where $p(\mathbf{x}|\mathbf{w}) = \delta_w(\mathbf{x})$ reflects the identity map. In Equation 4.6, note that $p(\mathbf{z}|\mathbf{w})$ is linear Gaussian by the well-known conditioning formula for jointly Gaussian distributions. We then arrive at Equation 4.7, where it can be checked that in Equation 4.8 both parts, $p(\mathbf{z}|\mathbf{w})$ and $p(\mathbf{y}|\mathbf{x},\mathbf{z})$, are linear Gaussian, thus makes $p(\mathbf{y}|\mathbf{x},\mathbf{w})$ linear Gaussian. Finally, we use the reparameterization trick together with a Cholesky decomposition, as seen in Section 4.3.2, to turn $p(\mathbf{w})$ into a standard Gaussian $p(\mathbf{u})$, which makes $p(\mathbf{x}|\mathbf{u})$, as a composition of identity map and linear Gaussian also a linear Gaussian. Note that $p(\mathbf{y}|\mathbf{x},\mathbf{u})$ again is linear Gaussian by similar arguments. Last we use the reparameterization trick again to obtain $p(\mathbf{y}|\mathbf{x},\mathbf{u},\mathbf{v})$ where $\mathbf{V} \sim \mathcal{N}(\mathbf{0},\mathbf{I}_N)$. □

4.7.4 Proof of Theorem 3.4

Proof. The linear version of the reduced SCM in Equation 4.36 entails the following distributions over $\mathbf{x}$ and $\mathbf{y}$

$$p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}|\mathbf{a},BB^\top\right), \tag{4.37}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}\left(\mathbf{y}|\mathbf{c}+D\mathbf{x}+EB^{-1}(\mathbf{x}-\mathbf{a}),FF^\top\right), \tag{4.38}$$

$$p(\mathbf{y}|\mathrm{do}(\mathbf{x})) = \mathcal{N}\left(\mathbf{y}|\mathbf{c}+D\mathbf{x},EE^\top+FF^\top\right), \tag{4.39}$$

Comparing Equations 4.12, 4.13, 4.14 with 4.37, 4.38, 4.39 we immediately get the equations for the parameters:

$$\boldsymbol{\alpha} = \mathbf{a}, \tag{4.40}$$

$$\Sigma = BB^\top, \tag{4.41}$$

$$\boldsymbol{\gamma} + \Delta\mathbf{x} = \mathbf{c} + \left(D + EB^{-1}\right)\mathbf{x} - EB^{-1}\mathbf{a}, \tag{4.42}$$

$$\boldsymbol{\gamma} \overset{\mathbf{x}=\mathbf{0}}{=} \mathbf{c} - EB^{-1}\mathbf{a}, \tag{4.43}$$

$$\Pi = FF^\top, \tag{4.44}$$

$$\widetilde{\boldsymbol{\gamma}} = \mathbf{c}, \tag{4.45}$$

$$\widetilde{\Delta} = D, \tag{4.46}$$

$$\widetilde{\Pi} = EE^\top + FF^\top. \tag{4.47}$$

Substituting $\mathbf{a}, \mathbf{c}, D, FF^\top$ and then subtracting Equation 4.43 from 4.42 and solving for all $\mathbf{x}$ we get the constraints:

$$\Delta = \widetilde{\Delta} + EB^{-1}, \tag{4.48}$$

$$\boldsymbol{\gamma} = \widetilde{\boldsymbol{\gamma}} - EB^{-1}\boldsymbol{\alpha}, \tag{4.49}$$

$$\widetilde{\Pi} = \Pi + EE^\top. \tag{4.50}$$

With Equation 4.48 we see that $E = (\Delta - \widetilde{\Delta})B$, which we can just plug into Equations 4.49 and 4.50. Finally using Equation 4.41 to replace $BB^\top$ with Σ in Equation 4.50 will give the claim. □

4.7.5 Reduction with observed confounders

There are many scenarios where we are interested in estimating the conditional causal effect of interventions given additional covariates $\mathbf{C}$ that might confound treatment and outcome, for example when estimating the efficacy of a vaccine depending on age, weight or gender. We again consider a treatment variable $\mathbf{X} \in \mathscr{X} = \mathbb{R}^M$, an outcome variable $\mathbf{Y} \in \mathscr{Y} = \mathbb{R}^N$, and a set of K latent confounders $Z_1, \ldots, Z_K$ in arbitrary standard measurable spaces (e.g., $\mathbb{R}^d$ or discrete). In addition, let there be L observed confounders $C_1, \ldots, C_L$ of $\mathbf{X}$ and $\mathbf{Y}$, again in arbitrary standard measurable spaces. We allow for arbitrary causal relations and dependencies between the confounders. In the following, we summarize all observed confounders using a single variable $\mathbf{C} = (C_1, \ldots, C_L) \in \mathscr{C}$ and all latent confounders as $\mathbf{Z} = (Z_1, \ldots, Z_K) \in \mathscr{Z}$. We follow a similar sequence of steps as in Section 4.3.1 to derive a reduced causal model of the following form

$$p(\mathbf{x}, \mathbf{y}, \mathbf{w}, \mathbf{c}) = p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \mathbf{c})p(\mathbf{x}|\mathbf{w})p(\mathbf{w}|\mathbf{c})p(\mathbf{c}). \tag{4.51}$$

as illustrated in Figure 4.2 (a–d). At every step, the Bayesian network is observationally equivalent to the ones before and also interventionally equivalent for interventions on $\mathbf{X}$ and $\mathbf{Y}$.

We can now use a similar approach as in Section 4.3.2, and in addition marginalize out $\mathbf{W}$ as seen in Figure 4.2 (g), to convert the causal Bayesian Network into an SCM with structural equations of the form given below

$$\mathbf{X} = F(\mathbf{U}, \mathbf{C}), \tag{4.52}$$

$$\mathbf{Y} = G(\mathbf{V}, \mathbf{X}, \mathbf{U}, \mathbf{C}), \tag{4.53}$$

where $\mathbf{U} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_M) \perp\!\!\!\perp \mathbf{V} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}_N\right)$ and F and G are two deterministic maps. This is illustrated in Figure 4.2 (e–h). Again, at every step, the Bayesian network is observationally equivalent to the ones before and also interventionally equivalent for interventions on $\mathbf{X}$ and $\mathbf{Y}$.

4.7.6 Linear experiment

We now show the capabilities of our flow model to learn the model parameters jointly from observational and interventional data. Throughout this experiment we assume $x, y \in \mathbb{R}$. We generate training, validation and test data using the following linear SCM

Observational

$$u \sim \mathcal{N}(0,1) \perp\!\!\!\perp v \sim \mathcal{N}(0,1) \tag{4.54}$$

$$x_o = 2 \cdot u + 1 \tag{4.55}$$

$$y_o = 1.5 \cdot v - x_o - 3 \cdot u + 2 \tag{4.56}$$

Interventional

$$u \sim \mathcal{N}(0,1) \perp\!\!\!\perp v \sim \mathcal{N}(0,1) \tag{4.57}$$

$$x_i \sim \mathcal{N}(0,1) \tag{4.58}$$

$$y_i = 1.5 \cdot v - x_i - 3 \cdot u + 2 \tag{4.59}$$

Since we know that the data is generated by a linear SCM we choose the transformations in our flow model to be linear as well

$$u = f_{a,b}(x) = a \cdot x + b \tag{4.60}$$

$$x = f_{a,b}^{-1}(u) = \frac{1}{a} \cdot (u - b) \tag{4.61}$$

$$v = g_{x,u;c,d,e,f}(y) = c \cdot y + d \cdot x + e \cdot u + f \tag{4.62}$$

$$y = g_{x,u;c,d,e,f}^{-1}(v) = \frac{1}{c} \cdot (v - d \cdot x - e \cdot u - f), \tag{4.63}$$

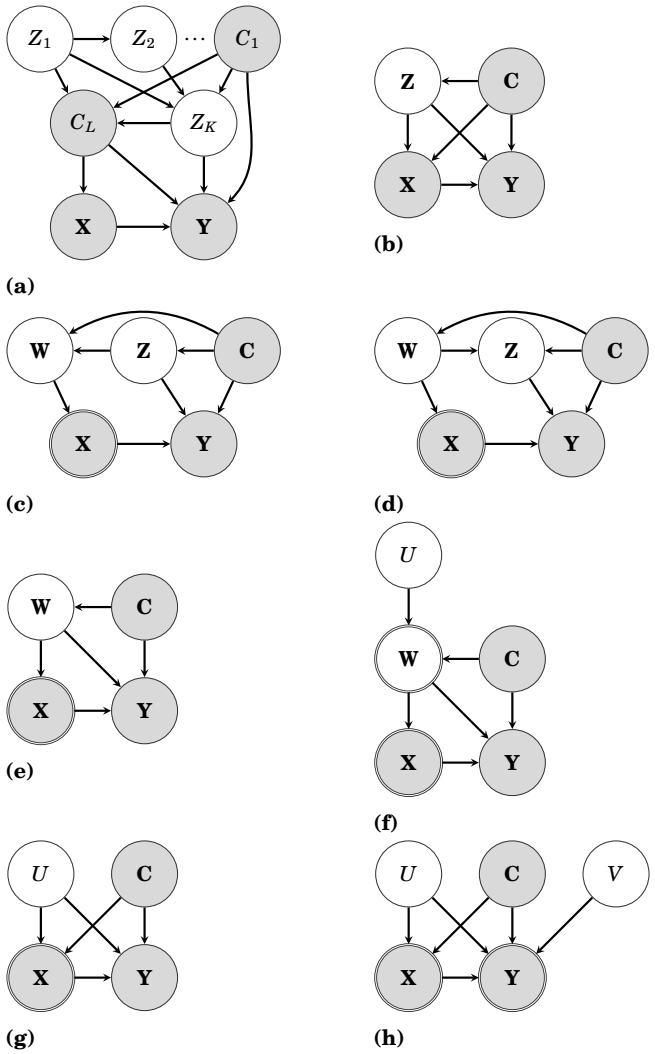


Figure 4.2. A graphical explanation of our reduction technique in the presence of both observed and latent confounders. (a) We assume a treatment variable $\mathbf{X} \in \mathbb{R}^M$, an outcome variable $\mathbf{Y} \in \mathbb{R}^N$, latent confounders $Z_1, \ldots, Z_K$, and observed confounders $C_1, \ldots, C_L$, with arbitrary causal and probabilistic relations between the confounders. (b) We combine the latent confounders into $\mathbf{Z} \in \mathcal{Z}$ and the observed confounders into $\mathbf{C} \in \mathcal{C}$, and factorize their joint distribution as $p(\mathbf{z} \mid \mathbf{c})p(\mathbf{c})$. (c) We create a copy of $\mathbf{X}$ called $\mathbf{W}$. (d) We refactorize $p(\mathbf{w}, \mathbf{z}, \mathbf{c})$ as $p(\mathbf{z} \mid \mathbf{w}, \mathbf{c})p(\mathbf{w} \mid \mathbf{c})p(\mathbf{c})$. (e) We marginalize over $\mathbf{Z}$. (f) We reparameterize $p(\mathbf{w} \mid \mathbf{c})$ using Theorem 4.3.2 as a deterministic function, introducing an independent noise variable $\mathbf{U}$. (g) We marginalize over $\mathbf{W}$. (h) We reparameterize $p(\mathbf{y} \mid \mathbf{x}, \mathbf{u}, \mathbf{c})$ with Theorem 4.3.2 as a deterministic function, introducing an independent noise variable $\mathbf{V}$. Note that at every step (a–h) the models remain observationally equivalent (i.e., $p(\mathbf{c}, \mathbf{x}, \mathbf{y})$ is invariant), and also interventionally equivalent with respect to interventions on $\mathbf{X}$ and $\mathbf{Y}$ (i.e., $p(\mathbf{y}, \mathbf{c} \mid \mathrm{do}(\mathbf{x}))$ and $p(\mathbf{x}, \mathbf{c} \mid \mathrm{do}(\mathbf{y}))$ are invariant).

in this case the volume terms in Equation 4.22 are simply given by

$$\left| \frac{\partial f_{a,b}(x)}{\partial x} \right| = |a| \tag{4.64}$$

$$\left| \frac{\partial g_{x,u;c,d,e,f}(y)}{\partial y} \right| = |c|\,. \tag{4.65}$$

Given a dataset consisting of observational and interventional data we can optimize the following loss

$$loss = \sum_{o=1}^{N_O} -\log p(x_o, y_o) + \alpha \sum_{i=1}^{N_I} -\log p(\mathrm{do}(x_i), y_i), \tag{4.66}$$

where

$$\begin{aligned} \log p(x_o, y_o) = \log p_V(c \cdot y_o + d \cdot x_o + e \cdot u + f) \\ + \log|c| + \log p_U(a \cdot x_o + b) + \log|a|\,, \end{aligned} \tag{4.67}$$

and

$$\begin{aligned} \log p(\mathrm{do}(x_i), y_i) = \log \int p_V(c \cdot y_i + d \cdot x_i + e \cdot u + f)|c|\, p(u) du \\ + \log p(x_i). \end{aligned} \tag{4.68}$$

We choose $\alpha = N_O/N_I$. In Equation 4.67, we now use $u = f_{a,b}(x) = a \cdot x + b$ to impute u. Resulting in

$$\begin{aligned} \log p(x_o, y_o) = \log p_V(c \cdot y_o + d \cdot x_o + e \cdot (a \cdot x_o + b) + f) \\ + \log|c| + \log p_U(a \cdot x_o + b) + \log|a|\,. \end{aligned} \tag{4.69}$$

In Figure 4.3, we show the training data set (left) and samples from our flow model trained with 100 observational samples and 100 interventional samples (right). Our flow model correctly finds the parameters used in the SCM used to generate the data. After training, our flow model can generate both observational and interventional samples, as described in Section 4.4.3.

4.7.7 Background: Normalizing Flows

Normalizing flows are based on the idea of transforming samples from a simple distribution into samples from a complex distribution using the change of variable formula [Rezende and Mohamed, 2015, Tabak and Turner, 2013]:

$$p(\mathbf{x}) = p_{\mathbf{Z}}(f(\mathbf{x})) \left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) \right|, \tag{4.70}$$

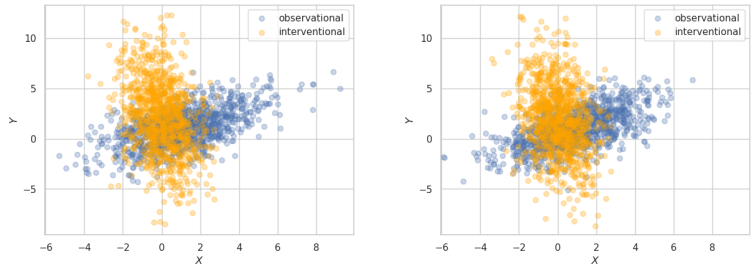


Figure 4.3. Samples from linear Gaussian model. Left: 1000 observational and 1000 interventional samples generated from the linear SCM in Section 4.7.6. Right: 1000 observational and 1000 interventional samples generated from our flow model trained with 100 observational and 100 interventional samples.

where $\mathbf{z} = f(\mathbf{x})$ is a bijective map $f : \mathcal{X} \to \mathcal{Z}$, $p_{\mathbf{Z}}(\mathbf{z})$ a simple prior distribution, and $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ the Jacobian with respect to $\mathbf{x}$. The transformation $f(\mathbf{x})$ is commonly composed of K transformations $f(\mathbf{x}) = f_K \circ \cdots \circ f_1(\mathbf{x})$ to increase the overall expressivity of $f(\mathbf{x})$. The choice of $f(\mathbf{x})$ is restricted by the computational complexity of calculating the Jacobian $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$. In recent years, a multitude of transformations with easy to compute Jacobians have been developed, for an overview see Kobyzev et al. [2020], Papamakarios et al. [2021].

In this paper we will use neural spline flows [Durkan et al., 2019, Dolatabadi et al., 2020]. Neural spline flows have two major advantages: 1. A better functional flexibility than affine transformations ($\mathbf{y} = \mathbf{s}\mathbf{x} + \mathbf{t}$), 2. A numerically stable, analytic inverse that has the same computational and space complexities as the forward operation. While Durkan et al. [2019] use quadratic, cubic, and rational quadratic functions whose inversion is done after solving polynomial equations, Dolatabadi et al. [2020] show that piecewise linear rational splines can perform competitively with these methods without requiring a polynomial equation to be solved in the inversion. Because of its reduced computational cost, we will use linear rational splines throughout this paper.

Consider a set of monotonically increasing points $\{(x^{(k)}, y^{(k)})\}_{k=0}^{K}$ called knots and a set of derivatives at each of the points $\{d^{(k)}\}_{k=0}^{K}$. For each bin $[x^{(k)}, x^{(k+1)}]$ we want to find a linear rational function of the form $\frac{ax+b}{cx+d}$ that fit the given points and derivatives.

The values returned by Algorithm 1 are subsequentely used to express the following linear rational spline function

$$f(\boldsymbol{\phi}) = \begin{cases} \frac{w^{(k)} y^{(k)} (\lambda^{(k)} - \boldsymbol{\phi}) + w^{(m)} y^{(m)} \boldsymbol{\phi}}{w^{(k)} (\lambda^{(k)} - \boldsymbol{\phi}) + w^{(m)} \boldsymbol{\phi}} & 0 \le \boldsymbol{\phi} \le \lambda^{(k)} \\ \frac{w^{(m)} y^{(m)} (1 - \boldsymbol{\phi}) + w^{(k+1)} y^{(k+1)} (\boldsymbol{\phi} - \lambda^{(k)})}{w^{(m)} (1 - \boldsymbol{\phi}) + w^{(k+1)} (\boldsymbol{\phi} - \lambda^{(k)})} & \lambda^{(k)} \le \boldsymbol{\phi} \le 1 \end{cases} \tag{4.71}$$

where $\boldsymbol{\phi} = (x - x^{(k)})/(x^{(k+1)} - x^{(k)})$.

Spline flows have two hyperparameters, the boundary B of the interval $[-B, B]$ and the number of bins K. Outside of the interval $[-B, B]$, the

Algorithm 1 Fuhr and Kallay [1992] Linear Rational Spline Interpolation of Monotonic data in the interval $\left[x^{(k)}, x^{(k+1)}\right]$.

Input: $x^{(k)} < x^{(k+1)}$, $y^{(k)} < y^{(k+1)}$, $d^{(k)} > 0$, $d^{(k+1)} > 0$

1: set $w^{(k)} > 0$

2: set $0 < \lambda^{(k)} < 1$

3: $w^{(k)} = \sqrt{\frac{d^{(k)}}{d^{(k+1)}}} w^{(k)}$

4: $y^m = \frac{w^{(k)} y^{(k)} (1 - \lambda^{(k)}) + w^{(k+1)} y^{(k+1)} \lambda^{(k)}}{w^{(k)} (1 - \lambda^{(k)}) + w^{(k+1)} \lambda^{(k)}}$

5: $w^{(m)} = \left(\lambda^{(k)} w^{(k)} d^{(k)} + \left(1 - \lambda^{(k)}\right) w^{(k+1)} d^{(k+1)}\right) \frac{x^{(k+1)} - x^{(k)}}{y^{(k+1)} - y^{(k)}}$

Return: $\lambda^{(k)}, w^{(k)}, w^{(m)}, w^{(k+1)}, y^{(m)}$

identity function is used. Using Equation 4.70 we can update the parameters of the neural spline flow using maximum-likelihood estimation in combination with gradient descent. In the case where $\mathbf{x}$ has two or more dimensions, either coupling layers [Dinh et al., 2017] or autoregressive layers [Papamakarios et al., 2017] can be used.

At multiple points in this paper we are required to estimate conditional distributions, e.g. $p(\mathbf{y}|\mathbf{x})$, where we will use conditional normalizing flows to estimate conditional probabilities. We consider the mapping $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, which is bijective in $\mathcal{Y}$ and $\mathcal{Z}$, and a simple prior distribution $p_{\mathbf{Z}}(\mathbf{z})$. Again, using the change of variable formula we can express the conditional distributions $p(\mathbf{y}|\mathbf{x})$ as follows

$$p(\mathbf{y}|\mathbf{x}) = p_{\mathbf{Z}}(f_x(\mathbf{y})) \left| \det \left(\frac{\partial f_x(\mathbf{y})}{\partial \mathbf{y}} \right) \right|. \tag{4.72}$$

The conditional version of the linear rational spline transformation uses a neural network to predict the derivatives $\mathbf{d}$, width $\mathbf{w}$, height $\mathbf{h}$, and $\boldsymbol{\lambda}$ from $\mathbf{x}$: $\mathbf{w}, \mathbf{h}, \mathbf{d}, \boldsymbol{\lambda} = NN_{\boldsymbol{\theta}}(\mathbf{x})$.

4.7.8 Simulation details: Nonlinear experiments without observed confounders

The generation of observational and interventional samples follows Mooij et al. [2016]. Instead of using Gaussian processes to model the causal mechanisms, we use two randomly initialized neural networks, NN_1 and NN_2.

Sampling from a random distribution

We use the following steps to generate samples from a random distribution

1. $X \sim \mathcal{N}(0, 1)$

2. sort X in ascending order $= \vec{X}$

3. Sample from Gaussian Process: $F \sim \mathcal{N}(0, K_{\boldsymbol{\theta}}(\vec{X}) + \sigma^2 I)$, where for the kernel $K_{\boldsymbol{\theta}}$ we use the squared exponential covariance function with

automatic relevance determination kernel

4. use the trapezoidal rule to calculate the cumulative integral of $\exp(F)$, we obtain a vector G where each element G_i corresponds to $G_i = \int_{\overrightarrow{X_1}}^{\overrightarrow{X_i}} \exp(F(x))dx$

We will denote this whole sampling procedure by $G \sim \mathscr{RD}(\boldsymbol{\theta}, \sigma)$, where we sample $\boldsymbol{\theta}$ from a Gamma distribution $\Gamma(a,b)$ and set $\sigma = 0.0001$.

Generate observational and interventional data

1. Sample from latent variables

$$\boldsymbol{\theta}_{E_X} \sim \Gamma(a_{E_X}, b_{E_X}), \tag{4.73}$$

$$\boldsymbol{\theta}_{E_Y} \sim \Gamma(a_{E_Y}, b_{E_Y}), \tag{4.74}$$

$$\boldsymbol{\theta}_{\mathbf{Z}} \sim \Gamma(a_{\mathbf{Z}}, b_{\mathbf{Z}}), \tag{4.75}$$

$$E_X \sim \mathscr{RD}(\boldsymbol{\theta}_{E_X}, \sigma), \tag{4.76}$$

$$E_Y \sim \mathscr{RD}(\boldsymbol{\theta}_{E_Y}, \sigma), \tag{4.77}$$

$$\mathbf{Z} \sim \mathscr{RD}(\boldsymbol{\theta}_{\mathbf{Z}}, \sigma). \tag{4.78}$$

2. Generate $X_{\text{observational}}$

$$X_{\text{observational}} = NN_1(E_X, \mathbf{Z}). \tag{4.79}$$

3. Normalize $X_{\text{observational}}$

$$X_{\text{observational}} = \frac{X_{\text{observational}} - \mathbb{E}\left[X_{\text{observational}}\right]}{\sqrt{\mathbb{V}[X_{\text{observational}}]}}. \tag{4.80}$$

4. Generate $Y_{\text{observational}}$

$$Y_{\text{observational}} = NN_2(X_{\text{observational}}, E_Y, \mathbf{Z}). \tag{4.81}$$

5. Sample from latent variables

$$E_Y \sim \mathscr{RD}(\boldsymbol{\theta}_{E_Y}, \sigma) \tag{4.82}$$

$$\mathbf{Z} \sim \mathscr{RD}(\boldsymbol{\theta}_{\mathbf{Z}}, \sigma) \tag{4.83}$$

6. Generate $X_{\text{interventional}}$

$$\boldsymbol{\theta}_X \sim \Gamma(a_X, b_X), \tag{4.84}$$

$$X_{\text{interventional}} \sim \mathscr{RD}(\boldsymbol{\theta}_X, \sigma). \tag{4.85}$$

7. Normalize $X_{\text{interventional}}$

$$X_{\text{interventional}} = \frac{X_{\text{interventional}} - \mathbb{E}\left[X_{\text{interventional}}\right]}{\sqrt{\mathbb{V}[X_{\text{interventional}}]}}. \tag{4.86}$$

8. Generate $Y_{\text{interventional}}$

$$Y_{\text{inter}} = NN_2(X_{\text{inter}}, E_Y, \mathbf{Z}). \tag{4.87}$$

9. Generate noise

$$\epsilon_{x,\text{observational}} \sim \mathcal{N}(0,1), \tag{4.88}$$

$$\epsilon_{x,\text{interventional}} \sim \mathcal{N}(0,1), \tag{4.89}$$

$$\boldsymbol{\theta}_{\epsilon_x} \sim \Gamma(a_{\epsilon_x}, b_{\epsilon_x}), \tag{4.90}$$

$$\epsilon_{y,\text{observational}} \sim \mathcal{N}(0,1), \tag{4.91}$$

$$\epsilon_{y,\text{interventional}} \sim \mathcal{N}(0,1), \tag{4.92}$$

$$\boldsymbol{\theta}_{\epsilon_y} \sim \Gamma(a_{\epsilon_y}, b_{\epsilon_y}). \tag{4.93}$$

10. Add noise

$$X'_{\text{observational}} = X_{\text{observational}} + \boldsymbol{\theta}_{\epsilon_x} \epsilon_{x,\text{observational}}, \tag{4.94}$$

$$X'_{\text{interventional}} = X_{\text{interventional}} + \boldsymbol{\theta}_{\epsilon_x} \epsilon_{x,\text{interventional}}, \tag{4.95}$$

$$Y'_{\text{observational}} = Y_{\text{observational}} + \boldsymbol{\theta}_{\epsilon_y} \epsilon_{y,\text{observational}}, \tag{4.96}$$

$$Y'_{\text{interventional}} = Y_{\text{interventional}} + \boldsymbol{\theta}_{\epsilon_y} \epsilon_{y,\text{interventional}}. \tag{4.97}$$

11. Normalize Y jointly

$$Y' = [Y'_{\text{observational}}, Y'_{\text{interventional}}], \tag{4.98}$$

$$Y'_{\text{observational}} = \frac{Y'_{\text{observational}} - \mathbb{E}\left[Y'\right]}{\sqrt{\mathbb{V}[Y']}}, \tag{4.99}$$

$$Y'_{\text{interventional}} = \frac{Y'_{\text{interventional}} - \mathbb{E}\left[Y'\right]}{\sqrt{\mathbb{V}[Y']}}. \tag{4.100}$$

The two neural networks NN_1 and NN_2 are Multi-layer perceptrons with a single hidden layer. The hidden layer contains 1024 units. The input layer and the hidden layer use a ReLU activation function. The weights and biases for both neural networks are uniformly sampled from the interval $[-1,1]$. We choose the other simulation parameters as follows: $a_{E_X} = a_{E_Y} = a_Z = a_X = 5$, $a_{\epsilon_x} = a_{\epsilon_y} = 2$, $b_{E_X} = b_{E_Y} = b_Z = b_X = b_{\epsilon_x} = b_{\epsilon_y} = 0.1$, $\sigma = 0.0001$

4.7.9 Simulation details: Nonlinear experiments with observed confounders

In order to simulate data with additional observed confounders, we first generate $\mathbf{C}$

$$\boldsymbol{\theta}_C = \Gamma(a_C, b_C), \tag{4.101}$$

$$\mathbf{C} \sim \mathscr{RD}(\boldsymbol{\theta}_C, \sigma), \tag{4.102}$$

where $a_C = 10$ and $b_C = 1$. In addition, we modify steps 2,4 and 8 as follows

$$X_{\text{observational}} = NN_1(E_X, \mathbf{Z}, \mathbf{C}), \tag{4.103}$$

$$Y_{\text{observational}} = NN_2(X_{\text{observational}}, E_Y, \mathbf{Z}, \mathbf{C}), \tag{4.104}$$

$$Y_{\text{inter}} = NN_2(X_{\text{inter}}, E_Y, \mathbf{Z}, \mathbf{C}). \tag{4.105}$$

4.7.10 Nonlinear experiment results without observed confounders

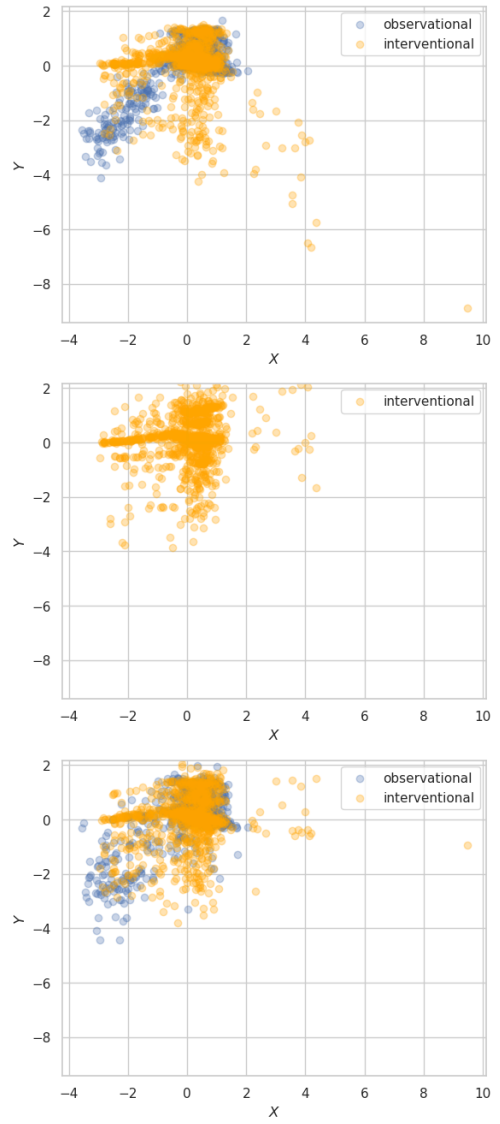


Figure 4.4. Dataset 1: Interventional and observational samples. Top: Observational and interventional training samples. Center: Interventional samples from a flow model trained with 50 interventional samples. Bottom: Observational and interventional samples from a flow model trained with 50 interventional samples and 1000 observational samples. The samples are generated as described in Section 4.4.3.

Figure 4.5. Dataset 1: Performances measured in terms of negative log-likelihood on the observational and the interventional test sets, respectively. Top: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 interventional samples from the test set. Bottom: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 observational samples from the test set. We report the mean and standard error for ten runs of each experiment.

Dataset 1: # of confounders = 1, random seed = 6

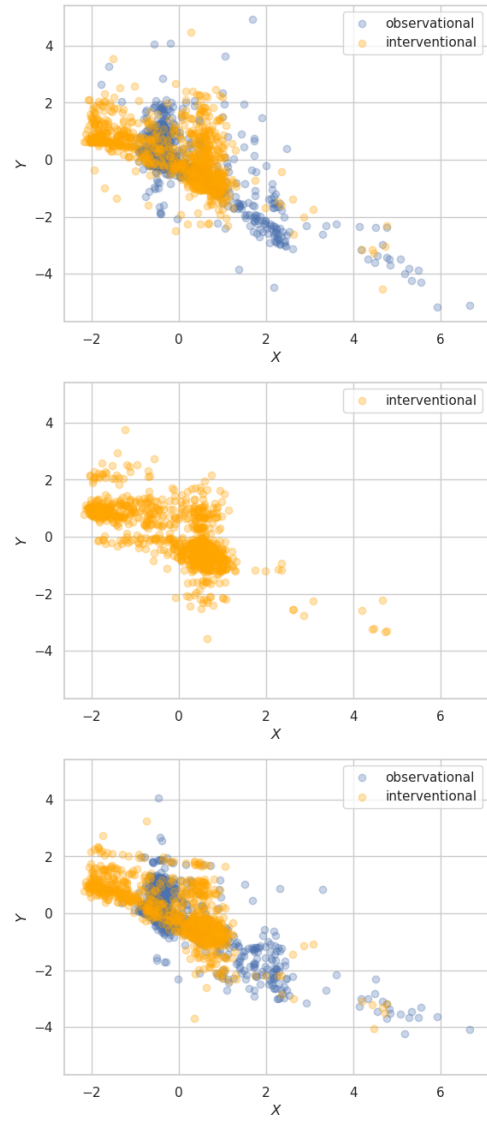


Figure 4.6. Dataset 2: Interventional and observational samples. Top: Observational and interventional training samples. Center: Interventional samples from a model trained with 50 interventional samples. Bottom: Observational and interventional samples from a model trained with 50 interventional samples and 1000 observational samples. The samples are generated as described in Section 4.4.3.

Figure 4.7. Dataset 2: Performances measured in terms of negative log-likelihood on the observational and the interventional test sets, respectively. Top: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 interventional samples from the test set. Bottom: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 observational samples from the test set. We report the mean and standard error for ten runs of each experiment.

Dataset 2: # of confounders = 1, random seed = 8

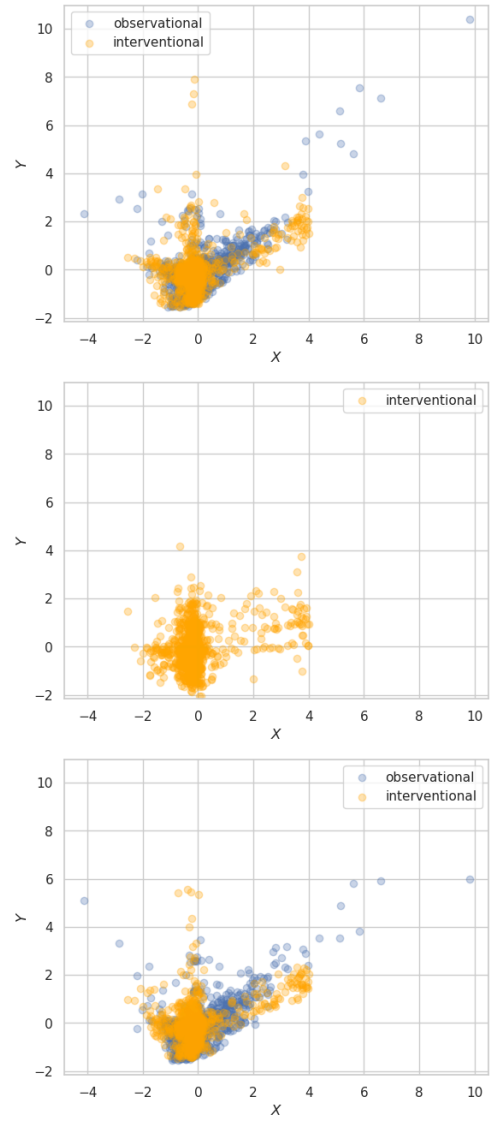


Figure 4.8. Dataset 3: Interventional and observational samples. Top: Observational and interventional training samples. Center: Interventional samples from a model trained with 50 interventional samples. Bottom: Observational and interventional samples from a model trained with 50 interventional samples and 1000 observational samples. The samples are generated as described in Section 4.4.3.

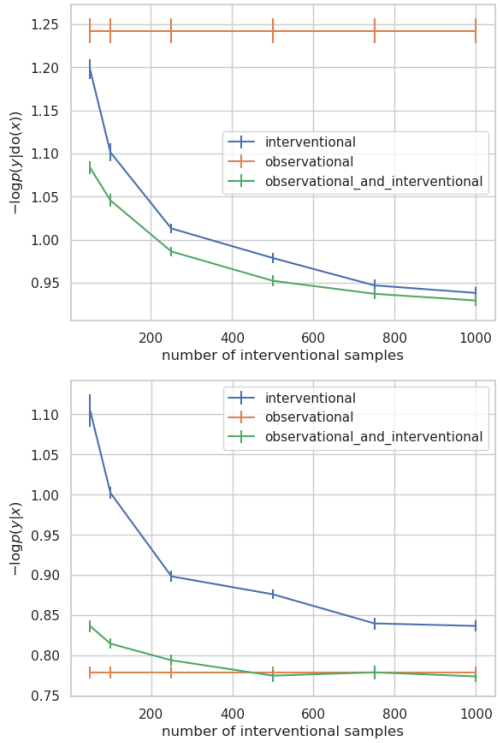


Figure 4.9. Dataset 3: Performances measured in terms of negative log-likelihood on the observational and the interventional test sets, respectively. Top: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 interventional samples from the test set. Bottom: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 observational samples from the test set. We report the mean and standard error for ten runs of each experiment.

Dataset 3: # of confounders = 2, random seed = 7

4.7.11 Dataset 4: 3 confounders, random seed = 1

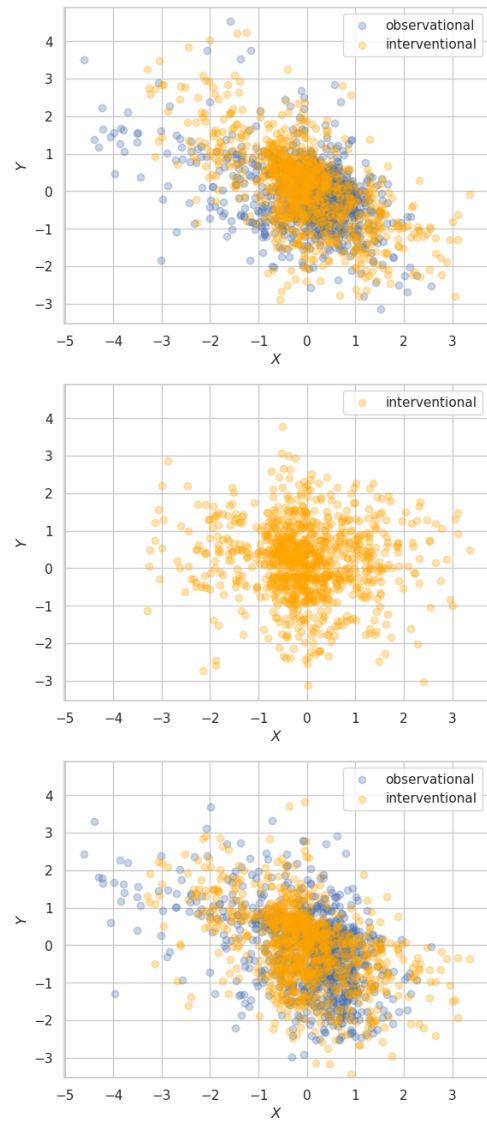


Figure 4.10. Dataset 4: Interventional and observational samples. Top: Observational and interventional training samples. Center: Interventional samples from a model trained with 50 interventional samples. Bottom: Observational and interventional samples from a model trained with 50 interventional samples and 1000 observational samples. The samples are generated as described in Section 4.4.3.

Figure 4.11. Dataset 4: Performances measured in terms of negative log-likelihood on the observational and the interventional test sets, respectively. Top: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 interventional samples from the test set. Bottom: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 observational samples from the test set. We report the mean and standard error for ten runs of each experiment.

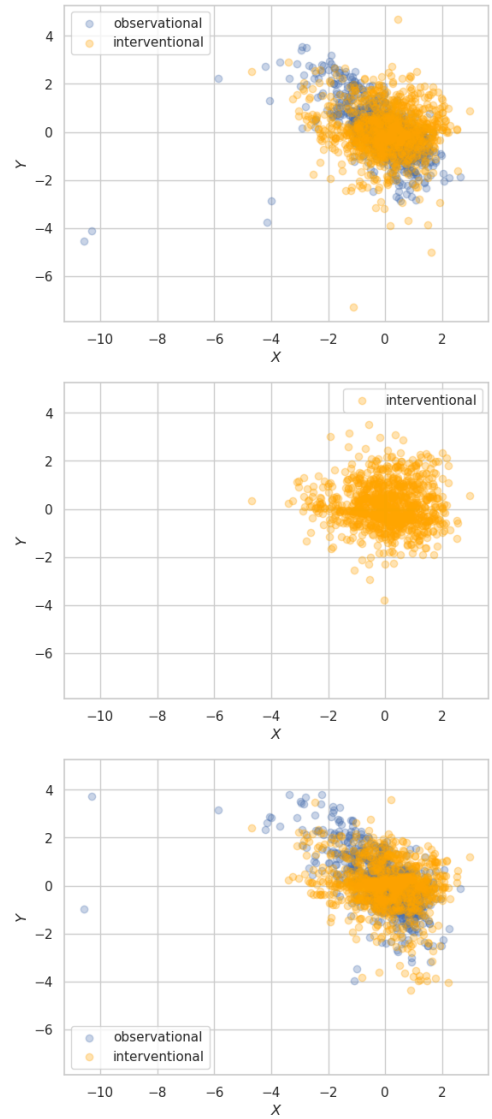


Figure 4.12. Dataset 5: Interventional and observational samples. Top: Observational and interventional training samples. Center: Interventional samples from a model trained with 50 interventional samples. Bottom: Observational and interventional samples from a model trained with 50 interventional samples and 1000 observational samples. The samples are generated as described in Section 4.4.3.

Figure 4.13. Dataset 5: Performances measured in terms of negative log-likelihood on the observational and the interventional test sets, respectively. Top: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 interventional samples from the test set. Bottom: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 observational samples from the test set. We report the mean and standard error for ten runs of each experiment.

Dataset 5: # of confounders = 4, random seed = 0

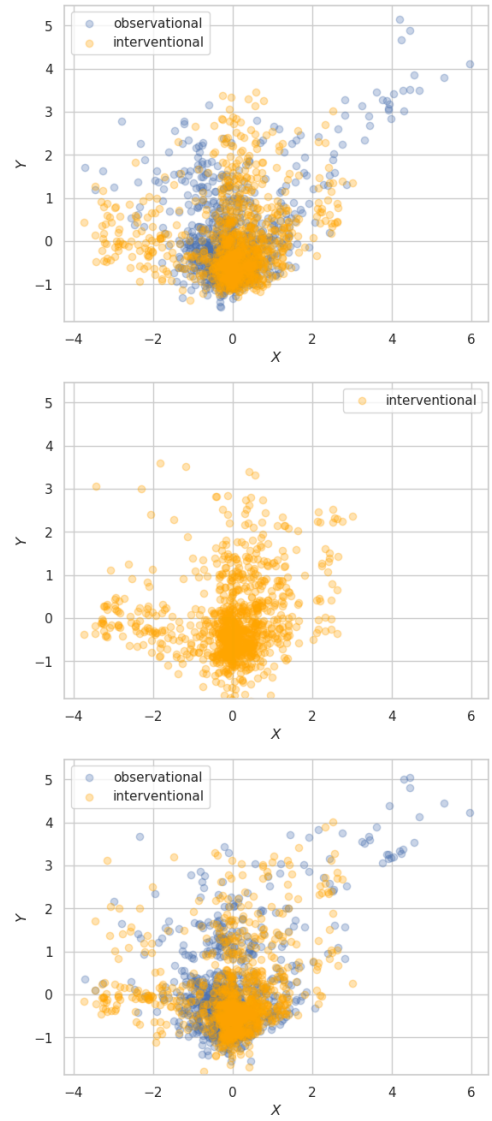


Figure 4.14. Dataset 6: Interventional and observational samples. Top: Observational and interventional training samples. Center: Interventional samples from a model trained with 50 interventional samples. Bottom: Observational and interventional samples from a model trained with 50 interventional samples and 1000 observational samples. The samples are generated as described in Section 4.4.3.

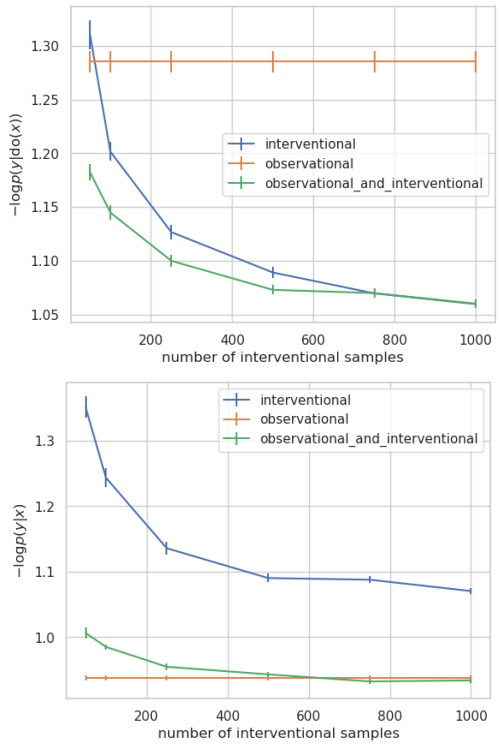


Figure 4.15. Dataset 6: Performances measured in terms of negative log-likelihood on the observational and the interventional test sets, respectively. Top: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 interventional samples from the test set. Bottom: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 observational samples from the test set. We report the mean and standard error for ten runs of each experiment.

Dataset 6: # of confounders = 4, random seed = 7

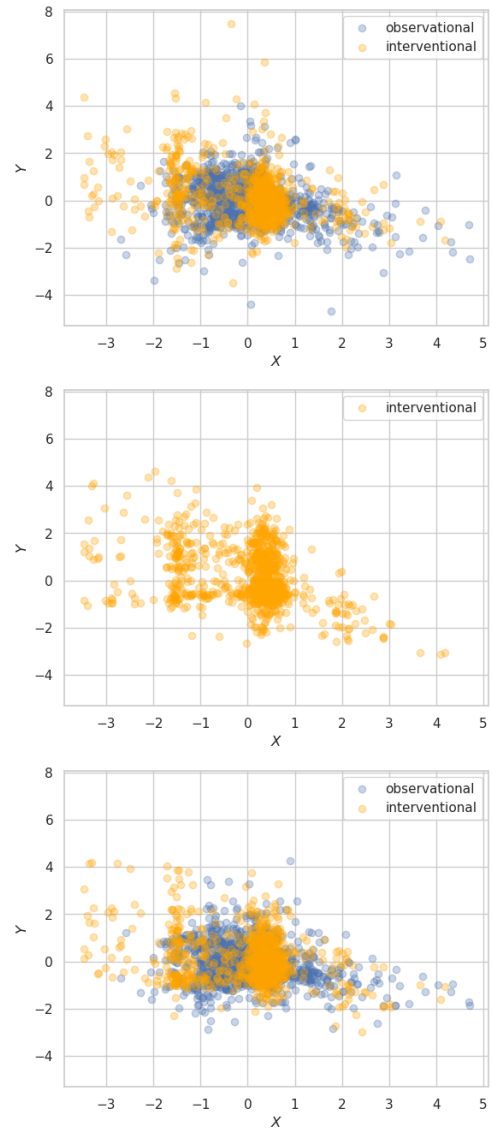


Figure 4.16. Dataset 7: Interventional and observational samples. Top: Observational and interventional training samples. Center: Interventional samples from a model trained with 50 interventional samples. Bottom: Observational and interventional samples from a model trained with 50 interventional samples and 1000 observational samples. The samples are generated as described in Section 4.4.3.

Figure 4.17. Dataset 7: Performances measured in terms of negative log-likelihood on the observational and the interventional test sets, respectively. Top: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 interventional samples from the test set. Bottom: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 observational samples from the test set. We report the mean and standard error for ten runs of each experiment.

Dataset 7: # of confounders = 5, random seed = 5

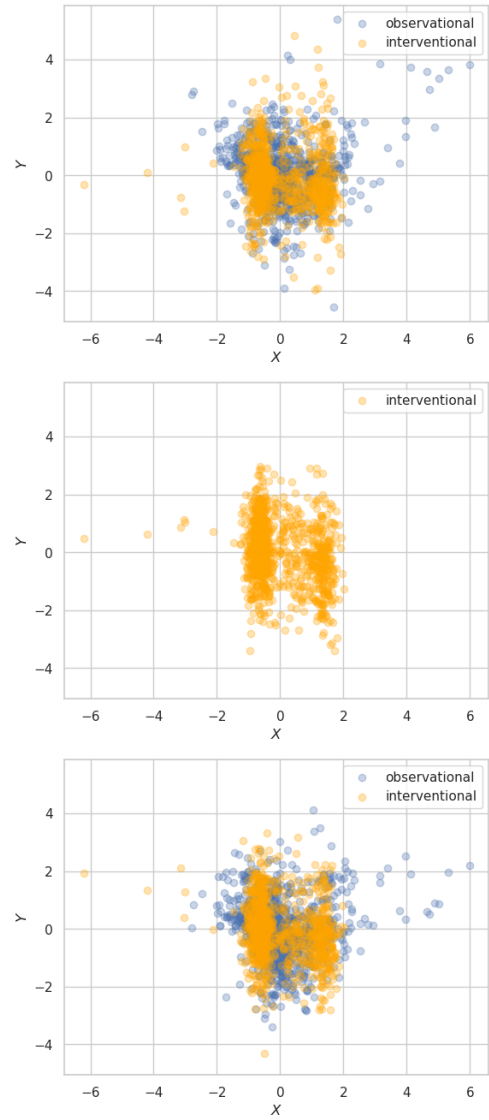


Figure 4.18. Dataset 8: Interventional and observational samples. Top: Observational and interventional training samples. Center: Interventional samples from a model trained with 50 interventional samples. Bottom: Observational and interventional samples from a model trained with 50 interventional samples and 1000 observational samples. The samples are generated as described in Section 4.4.3.

Figure 4.19. Dataset 8: Performances measured in terms of negative log-likelihood on the observational and the interventional test sets, respectively. Top: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 interventional samples from the test set. Bottom: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 observational samples from the test set. We report the mean and standard error for ten runs of each experiment.

Dataset 8: # of confounders = 5, random seed = 9

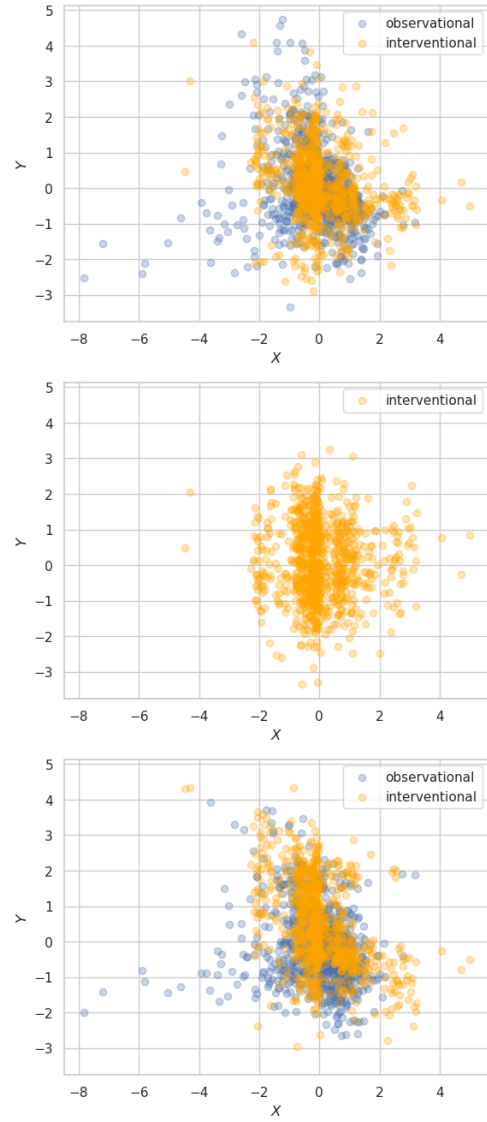


Figure 4.20. Dataset 9: Interventional and observational samples. Top: Observational and interventional training samples. Center: Interventional samples from a model trained with 50 interventional samples. Bottom: Observational and interventional samples from a model trained with 50 interventional samples and 1000 observational samples. The samples are generated as described in Section 4.4.3.

Figure 4.21. Dataset 9: Performances measured in terms of negative log-likelihood on the observational and the interventional test sets, respectively. Top: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 interventional samples from the test set. Bottom: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 observational samples from the test set. We report the mean and standard error for ten runs of each experiment.

Dataset 9: # of confounders = 7, random seed = 0

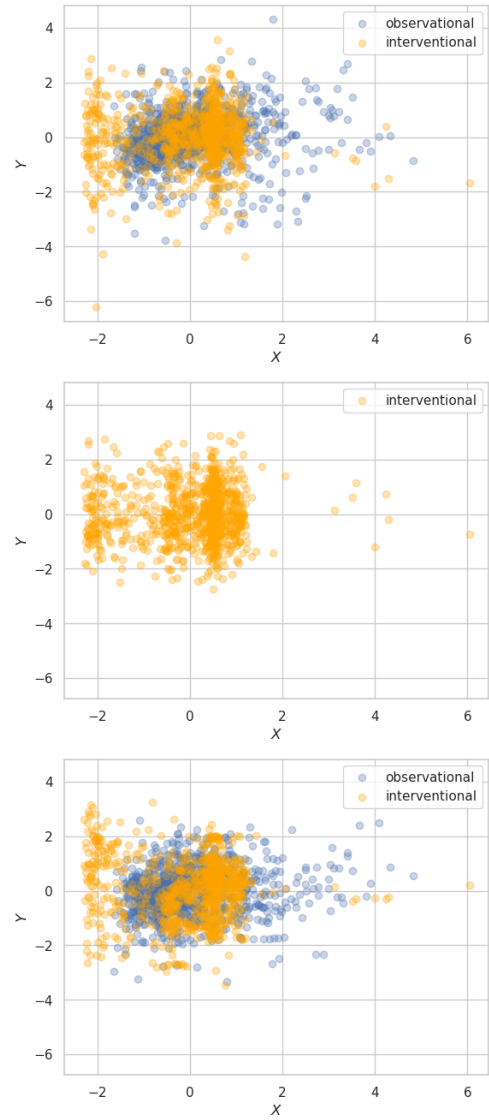


Figure 4.22. Dataset 10: Interventional and observational samples. Top: Observational and interventional training samples. Center: Interventional samples from a model trained with 50 interventional samples. Bottom: Observational and interventional samples from a model trained with 50 interventional samples and 1000 observational samples. The samples are generated as described in Section 4.4.3.

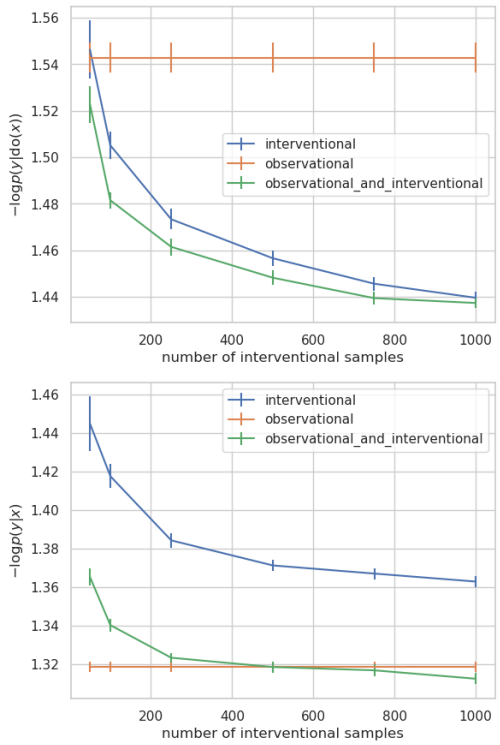


Figure 4.23. Dataset 10: Performances measured in terms of negative log-likelihood on the observational and the interventional test sets, respectively. Top: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 interventional samples from the test set. Bottom: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 observational samples from the test set. We report the mean and standard error for ten runs of each experiment.

Dataset 10: # of confounders = 7, random seed = 5

4.7.12 Nonlinear experiment results with observed confounders

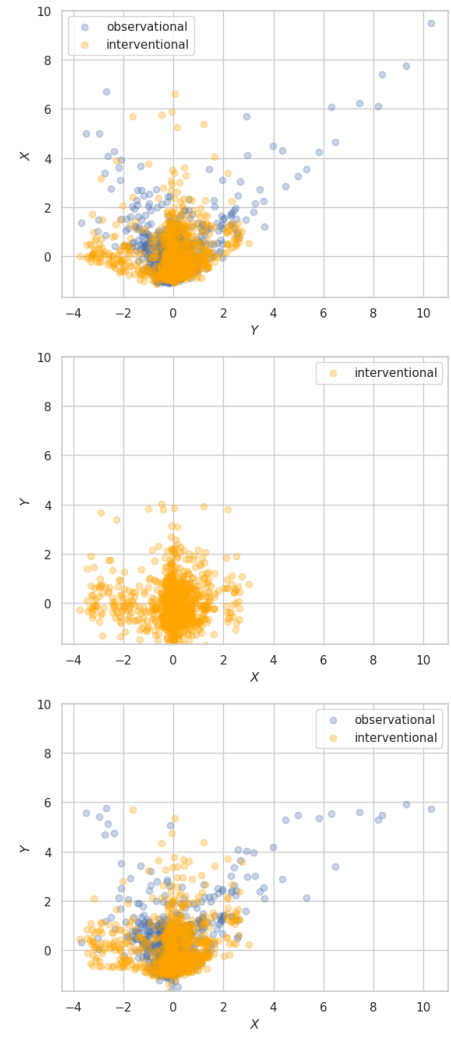


Figure 4.24. Dataset 11: Interventional and observational samples. Top: Observational and interventional training samples. Center: Interventional samples from a model trained with 50 interventional samples. Bottom: Observational and interventional samples from a model trained with 50 interventional samples and 1000 observational samples. The samples are generated as described in Section 4.4.3.

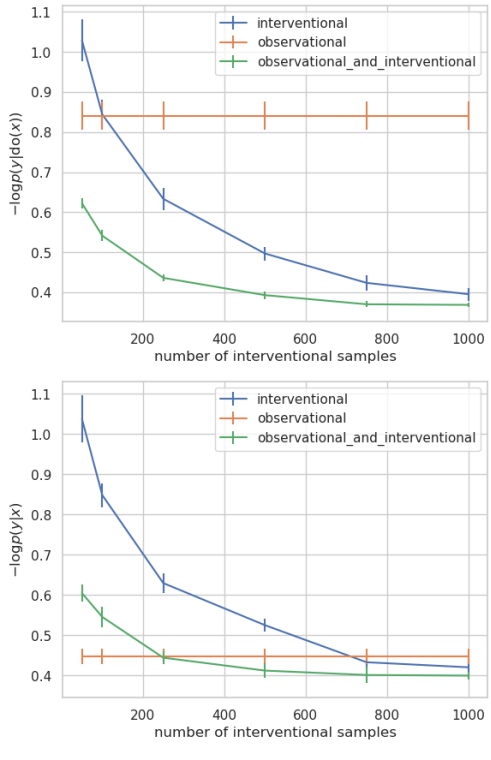


Figure 4.25. Dataset 11: Performances measured in terms of negative log-likelihood on the observational and the interventional test sets, respectively. Top: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 interventional samples from the test set. Bottom: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 observational samples from the test set. We report the mean and standard error for ten runs of each experiment.

Dataset 11: # of latent confounders = 1, # of observed confounders = 3, random seed = 7

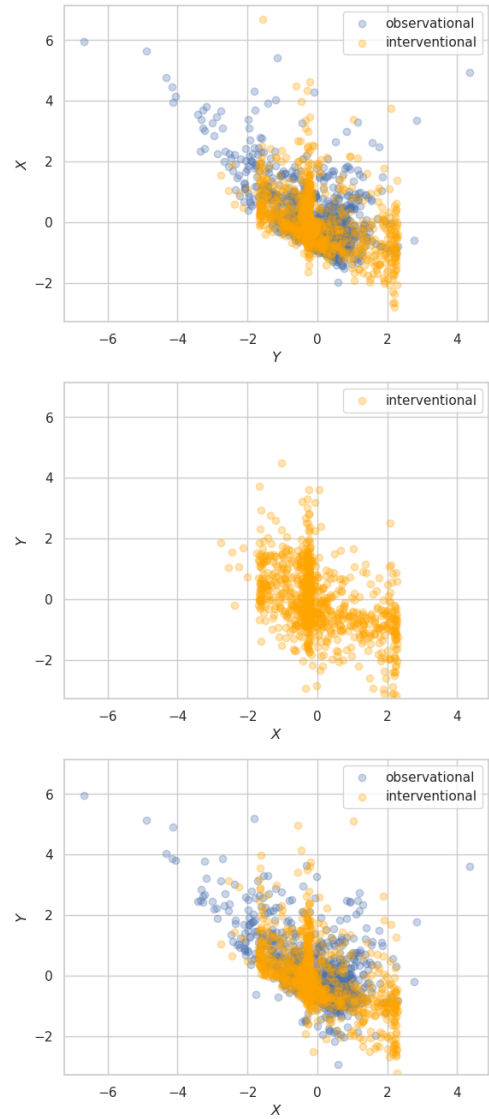


Figure 4.26. Dataset 12: Interventional and observational samples. Top: Observational and interventional training samples. Center: Interventional samples from a model trained with 50 interventional samples. Bottom: Observational and interventional samples from a model trained with 50 interventional samples and 1000 observational samples. The samples are generated as described in Section 4.4.3.

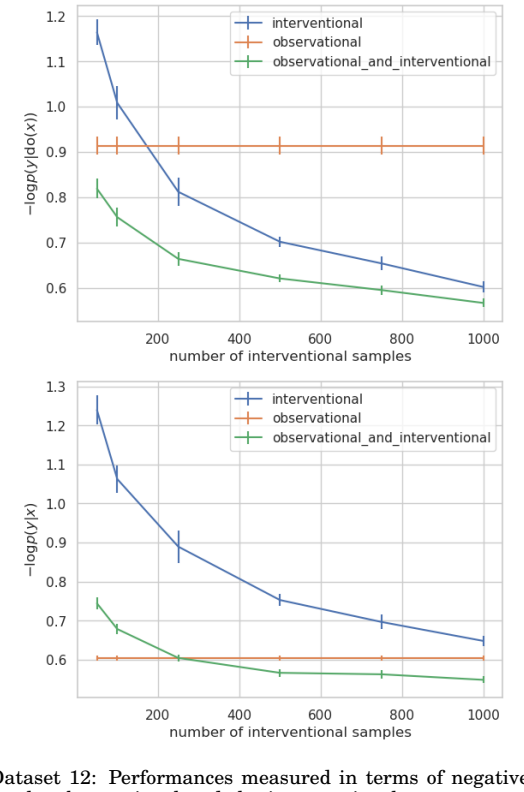


Figure 4.27. Dataset 12: Performances measured in terms of negative log-likelihood on the observational and the interventional test sets, respectively. Top: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 interventional samples from the test set. Bottom: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 observational samples from the test set. We report the mean and standard error for ten runs of each experiment.

Dataset 12: # of latent confounders = 1, # of observed confounders = 3, random seed = 9

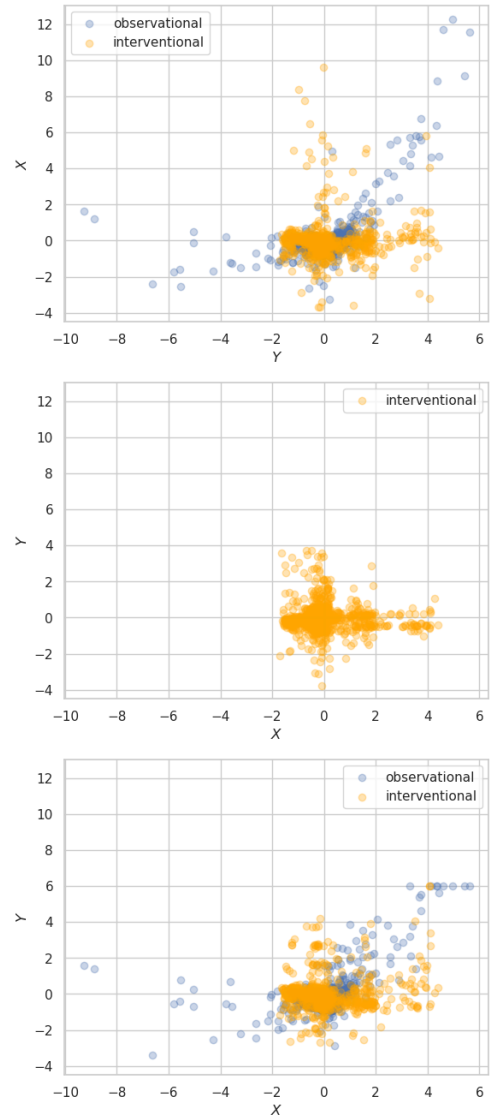


Figure 4.28. Dataset 13: Interventional and observational samples. Top: Observational and interventional training samples. Center: Interventional samples from a model trained with 50 interventional samples. Bottom: Observational and interventional samples from a model trained with 50 interventional samples and 1000 observational samples. The samples are generated as described in Section 4.4.3.

Figure 4.29. Dataset 13: Performances measured in terms of negative log-likelihood on the observational and the interventional test sets, respectively. Top: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 interventional samples from the test set. Bottom: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 observational samples from the test set. We report the mean and standard error for ten runs of each experiment.

Dataset 13: # of latent confounders = 2, # of observed confounders = 1, random seed = 0

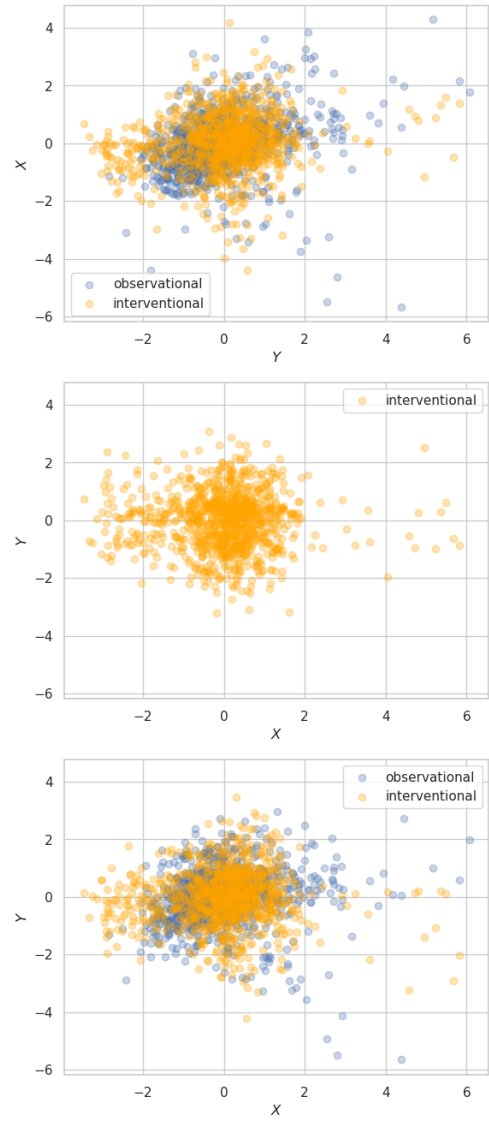


Figure 4.30. Dataset 14: Interventional and observational samples. Top: Observational and interventional training samples. Center: Interventional samples from a model trained with 50 interventional samples. Bottom: Observational and interventional samples from a model trained with 50 interventional samples and 1000 observational samples. The samples are generated as described in Section 4.4.3.

Figure 4.31. Dataset 14: Performances measured in terms of negative log-likelihood on the observational and the interventional test sets, respectively. Top: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 interventional samples from the test set. Bottom: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 observational samples from the test set. We report the mean and standard error for ten runs of each experiment.

Dataset 14: # of latent confounders = 3, # of observed confounders = 3, random seed = 5

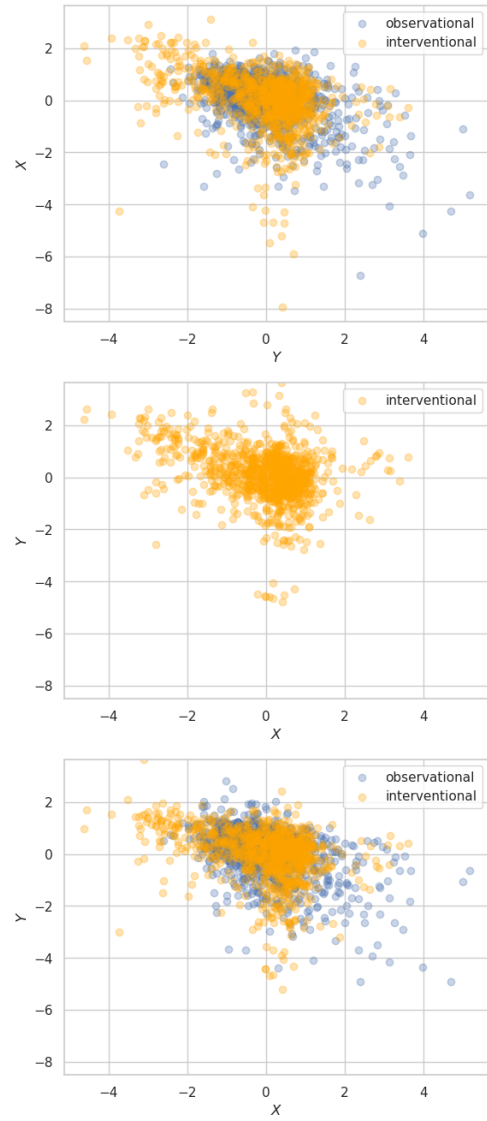


Figure 4.32. Dataset 15: Interventional and observational samples. Top: Observational and interventional training samples. Center: Interventional samples from a model trained with 50 interventional samples. Bottom: Observational and interventional samples from a model trained with 50 interventional samples and 1000 observational samples. The samples are generated as described in Section 4.4.3.

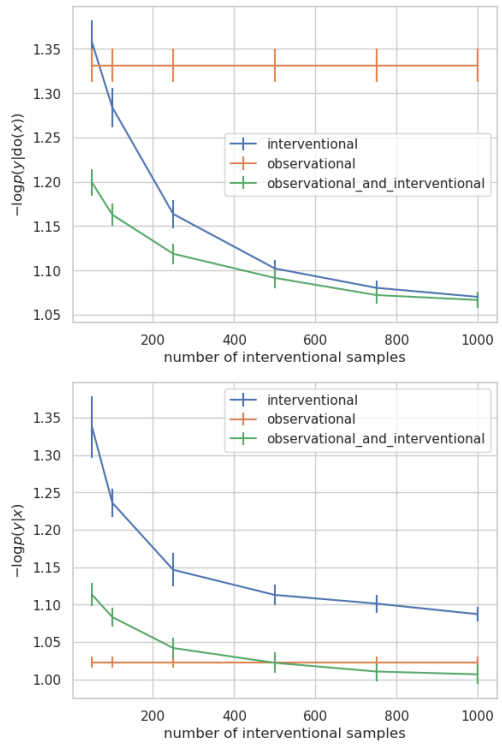


Figure 4.33. Dataset 15: Performances measured in terms of negative log-likelihood on the observational and the interventional test sets, respectively. Top: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 interventional samples from the test set. Bottom: Comparison of a flow model trained with 1000 observational samples, a flow model trained with 50, 100, 250, 500, 750, 1000 interventional samples, and a flow model trained with both 1000 observational samples and 50, 100, 250, 500, 750, 1000 interventional samples. All flow models are evaluated on 1000 observational samples from the test set. We report the mean and standard error for ten runs of each experiment.

Dataset 15: # of latent confounders = 4, # of observed confounders = 4, random seed = 2

5. DIVA: Domain Invariant Variational Autoencoders

In the last section of the main part of the thesis, we focus on what we consider the most challenging experimental setup. As in Chapter 4, we assume that we have no apriori knowledge about the symmetries of the data and the task. Therefore we cannot use invariant architecture or data augmentation techniques to train invariant machine learning models. In contrast to Chapter 4, we assume that we have no access to experimental data with known interventions either. Instead, we will solely rely on data from multiple domains. As seen in Section 1.2.2, we assume that there exist invariant causal mechanisms that generalize across all domains. Subsequently, we assume that latent features exist that are invariant to domain changes, so-called domain invariant features. We propose using an augmented VAE, see Section 1.3.1, to learn those domain invariant features. By learning disentangled features, we can obtain domain invariant features as a subset of the latent space. We demonstrate that we can learn features invariant to (semi)groups like color transformations and rotations without specifying the group symmetry before hands.

5.1 Introduction

Deep Neural Networks (DNNs) led to breakthroughs in various areas like computer vision and natural language processing. Despite their immense success, recent research shows that DNNs learn the bias present in the training data. As a result, they are not invariant to cues that are irrelevant to the actual task [Azulay and Weiss, 2019]. This leads to a dramatic performance decrease when tested on data from a different distribution with a different bias.

In domain generalization, the goal is to learn representations from a set of similar distributions called domains that can be transferred to a previously unseen domain during test time. A common motivating application, where domain generalization is crucial, is medical imaging [Blanchard et al., 2011, Muandet et al., 2013]. For instance, in digital histopathology, a

typical task is the classification of benign and malignant tissue. However, preparing a histopathology image includes tissue staining and scanning, which can significantly vary between hospitals. Moreover, the samples from a single patient could be preserved in different conditions [Ciompi et al., 2017]. As a result, each patient's data could be treated as a separate domain [Lafarge et al., 2017]. Another problem commonly encountered in medical imaging is class label scarcity. Annotating medical images is a highly time-consuming task that requires expert knowledge. However, obtaining domain labels is surprisingly cheap since hospitals generally store information about the patient (e.g., age and sex) and the medical equipment (e.g., manufacturer and settings). Therefore, we are interested in extending the domain generalization framework to deal with additional unlabeled data, as we hypothesize that it can help improve performance.

In this paper, we propose to tackle domain generalization via a new deep generative model that we refer to as the Domain Invariant Variational Autoencoder (DIVA). We extend the Variational AutoEncoder (VAE) framework [Kingma and Welling, 2013, Rezende et al., 2014] by introducing independent latent representations for a domain label d, a class label y and any residual variations in the input $\mathbf{x}$. Such partitioning of the latent space will encourage and guide the model to disentangle these sources of variation. Finally, by having a generative model, we can naturally handle the semi-supervised scenario, similarly to Kingma et al. [2014]. We evaluate our model on a version of the MNIST dataset where each domain corresponds to a specific rotation angle of the digits and on a Malaria Cell Images dataset where each domain corresponds to a different patient.

5.2 Definition of Domain Generalization

We follow the domain generalization definitions used in Muandet et al. [2013]. A domain is defined as a joint distribution $p(\mathbf{x}, y)$ on $\mathscr{X} \times \mathscr{Y}$, where $\mathscr{X}$ denotes the input space and $\mathscr{Y}$ denotes the output space. Let $\mathfrak{P}_{\mathscr{X}\times\mathscr{Y}}$ be the set of all domains. The training set consists of samples $\mathscr{S}$ taken from N domains, $\mathscr{S} = \{S^{(d=i)}\}_{i=1}^{N}$. Here, the ith domain $p^{(d=i)}(\mathbf{x}, y)$ is represented by n_i samples, $S^{(d=i)} = \{(\mathbf{x}_k^{(d=i)}, y_k^{(d=i)})\}_{k=1}^{n_i}$. Each of the N distributions $p^{(d=1)}(\mathbf{x}, y), \ldots, p^{(d=i)}(\mathbf{x}, y), \ldots, p^{(d=N)}(\mathbf{x}, y)$ are sampled from $\mathfrak{P}_{\mathscr{X}\times\mathscr{Y}}$. We further assume that $p^{(d=i)}(\mathbf{x}, y) \neq p^{(d=j)}(\mathbf{x}, y)$, therefore, the samples in $\mathscr{S}$ are non-i.i.d. During test time we are presented with samples $S^{(d=N+1)}$ from a previously unseen domain $p^{(d=N+1)}(\mathbf{x}, y)$. We are interested in learning representations that generalize from $p^{(d=1)}(\mathbf{x}, y), \ldots, p^{(d=N)}(\mathbf{x}, y)$ to this new domain. Training data are given as tuples $(d, \mathbf{x}, y)$ in the case of supervised data or as $(d, \mathbf{x})$ in the case of unsupervised data.

5.3 DIVA: Domain Invariant VAE

Assuming a perfectly disentangled latent space [Higgins et al., 2018], we hypothesize that there exists a latent subspace that is invariant to changes in d, i.e., it is domain invariant. We propose a generative model with three independent sources of variation; $\mathbf{z}_d$, which is domain specific, $\mathbf{z}_y$, which is class specific and finally $\mathbf{z}_x$, which captures any residual variations left in $\mathbf{x}$. While $\mathbf{z}_x$ keeps an independent Gaussian prior $p(\mathbf{z}_x)$, $\mathbf{z}_d$ and $\mathbf{z}_y$ have conditional priors $p_{\boldsymbol{\theta}_d}(\mathbf{z}_d|d)$, $p_{\boldsymbol{\theta}_y}(\mathbf{z}_y|y)$ with learnable parameters $\boldsymbol{\theta}_d, \boldsymbol{\theta}_y$. This will encourage information about the domain d and label y to be encoded into $\mathbf{z}_d$ and $\mathbf{z}_y$, respectively. Furthermore, as $\mathbf{z}_d$ and $\mathbf{z}_y$ are marginally independent by construction, we argue that the model will learn representations $\mathbf{z}_y$ that are invariant with respect to the domain d. All three of these latent variables are then used by a single decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_d, \mathbf{z}_x, \mathbf{z}_y)$ for the reconstruction of $\mathbf{x}$. Since we are interested in using neural networks to parameterize $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_d, \mathbf{z}_x, \mathbf{z}_y)$, exact inference will be intractable. For this reason, we perform amortized variational inference with an inference network [Kingma and Welling, 2013, Rezende et al., 2014], i.e., we employ a VAE-type framework. We introduce three separate encoders $q_{\boldsymbol{\phi}_d}(\mathbf{z}_d|\mathbf{x})$, $q_{\boldsymbol{\phi}_x}(\mathbf{z}_x|\mathbf{x})$ and $q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x})$ that serve as variational posteriors over the latent variables. Notice that we do not share their parameters as we empirically found that sharing parameters leads to a decreased generalization performance. For the prior and variational posterior distributions over the latent variables $\mathbf{z}_x, \mathbf{z}_d, \mathbf{z}_y$ we assume fully factorized Gaussians with parameters given as a function of their input. We coin the term Domain Invariant VAE (DIVA) for our overall model, which is depicted in Figure 5.1.

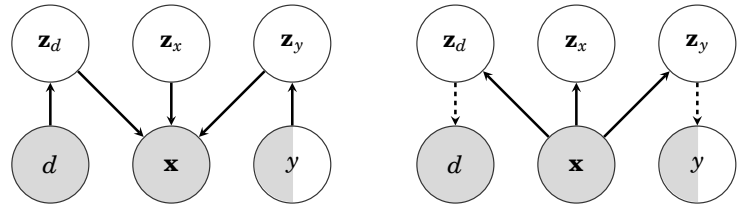


Figure 5.1. DAG of DIVA. Left: Generative model. Right: Inference model. A grey node means the variable is observed and a white node corresponds to a latent (unobserved) variable. Dashed arrows represent the auxiliary classifiers.

Given a specific dataset, all of the aforementioned parameters can be optimized by maximizing the following variational lower bound per input $\mathbf{x}$:

$$\begin{aligned}\mathscr{L}_s(d, \mathbf{x}, y) = \mathbb{E}_{q_{\boldsymbol{\phi}_d}(\mathbf{z}_d|\mathbf{x}) q_{\boldsymbol{\phi}_x}(\mathbf{z}_x|\mathbf{x}), q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_d, \mathbf{z}_x, \mathbf{z}_y)\right] \\ - \beta KL\left(q_{\boldsymbol{\phi}_d}(\mathbf{z}_d|\mathbf{x}) || p_{\boldsymbol{\theta}_d}(\mathbf{z}_d|d)\right) \\ - \beta KL\left(q_{\boldsymbol{\phi}_x}(\mathbf{z}_x|\mathbf{x}) || p(\mathbf{z}_x)\right)\end{aligned}$$

$$- \beta KL\left(q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x})||p_{\boldsymbol{\theta}_y}(\mathbf{z}_y|y)\right). \tag{5.1}$$

Notice that we have introduced a weighting term β. This is motivated by the β-VAE [Higgins et al., 2017] and serves as a constraint that controls the capacity of the latent spaces of DIVA. Larger values of β limit the capacity of each $\mathbf{z}$, and in the ideal case, each dimension of $\mathbf{z}$ captures one of the conditionally independent factors in $\mathbf{x}$.

To further encourage separation of $\mathbf{z}_d$ and $\mathbf{z}_y$ into domain- and class-specific information respectively, we add two auxiliary objectives. During training $\mathbf{z}_d$ is used to predict the domain d and $\mathbf{z}_y$ is used to predict the class y for a given input $\mathbf{x}$:

$$\begin{aligned}\mathscr{F}_{\text{DIVA}}(d,\mathbf{x},y) = \mathscr{L}_s(d,\mathbf{x},y) + \alpha_d \mathbb{E}_{q_{\boldsymbol{\phi}_d}(\mathbf{z}_d|\mathbf{x})}\left[\log q_{\boldsymbol{\omega}_d}(d|\mathbf{z}_d)\right] \\ + \alpha_y \mathbb{E}_{q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x})}\left[\log q_{\boldsymbol{\omega}_y}(y|\mathbf{z}_y)\right],\end{aligned} \tag{5.2}$$

where α_d, α_y are weighting terms for each of these auxiliary objectives. Since our main goal is a domain invariant classifier, during inference we only use the encoder $q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x})$ and the auxiliary classifier $q_{\boldsymbol{\omega}_y}(y|\mathbf{z}_y)$. For the prediction of the class y for a new input x, we use the mean of $\mathbf{z}_y$. Consequently, we regard the variational lower bound $\mathscr{L}_s(d,\mathbf{x},y)$ as a regularizer. Therefore, evaluating the marginal likelihood $p(\mathbf{x})$ of DIVA is outside the scope of this paper.

Locatello et al. [2019] and Dai and Wipf [2019] show that learning a disentangled representation, i.e., $q_{\boldsymbol{\phi}}(\mathbf{z}) = \prod_i q_{\boldsymbol{\phi}}(z_i)$, in an fully unsupervised fashion is impossible for arbitrary generative models. Inductive biases, e.g., some form of supervision or constraints on the latent space, are necessary to find a specific set of solutions that matches the true generative model. Consequently, DIVA is using domain labels d and class labels y in addition to input data $\mathbf{x}$ during training. Furthermore, we enforce the factorization of the marginal distribution of $\mathbf{z}$ in the following form: $q_{\boldsymbol{\phi}}(\mathbf{z}) = q_{\boldsymbol{\phi}_d}(\mathbf{z}_d) q_{\boldsymbol{\phi}_x}(\mathbf{z}_x) q_{\boldsymbol{\phi}_y}(\mathbf{z}_y)$, which prevents the impossibility described in Locatello et al. [2019]. We argue that the strong inductive biases in DIVA make it possible to learn disentangled representations that match the ground truth factors of interest, namely, the domain factors $\mathbf{z}_d$ and class factors $\mathbf{z}_y$.

In the Appendix, we perform multiple ablation studies that further justify the design of DIVA. We find that a partitioned latent space is indeed necessary to obtain good generalization performance. In line with the results in [Klys et al., 2018], we find that while omitting z_d is not leading to a significant decrease in generalization performance. It leads to a less interpretable model, see Section 5.5.2.

5.3.1 Semi-supervised DIVA

In Kingma et al. [2014] an extension to the VAE framework was introduced that allows to use labeled as well as unlabeled data during training. While Kingma et al. [2014] introduced a two step procedure, Louizos et al. [2016] present a way of optimizing the decoder of the VAE and the auxiliary classifier jointly. We use the latter approach to learn from supervised data $\{(d_n,\mathbf{x}_n,y_n)\}$ as well as from unsupervised data $\{(d_m,\mathbf{x}_m)\}$. Analogically to Louizos et al. [2016], we use $q_{\boldsymbol{\omega}_y}(y|\mathbf{z}_y)$ to impute y:

$$\begin{aligned}\mathscr{L}_u(d,\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}_d}(\mathbf{z}_d|\mathbf{x}) q_{\boldsymbol{\phi}_x}(\mathbf{z}_x|\mathbf{x}) q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_d,\mathbf{z}_x,\mathbf{z}_y)\right] \\ - \beta KL\left(q_{\boldsymbol{\phi}_d}(\mathbf{z}_d|\mathbf{x})||p_{\boldsymbol{\theta}_d}(\mathbf{z}_d|d)\right) \\ - \beta KL\left(q_{\boldsymbol{\phi}_x}(\mathbf{z}_x|\mathbf{x})||p(\mathbf{z}_x)\right) \\ + \beta \mathbb{E}_{q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x}) q_{\boldsymbol{\omega}_y}(y|\mathbf{z}_y)}\left[\log p_{\boldsymbol{\theta}_y}(\mathbf{z}_y|y) \log q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x})\right] \\ + \mathbb{E}_{q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x}) q_{\boldsymbol{\omega}_y}(y|\mathbf{z}_y)}\left[\log p(y) - \log q_{\boldsymbol{\omega}_y}(y|\mathbf{z}_y)\right],\end{aligned} \tag{5.3}$$

where we use Monte Carlo sampling with the reparametrization trick [Kingma and Welling, 2013] for the continuous latent variables $\mathbf{z}_d, \mathbf{z}_x, \mathbf{z}_y$ and explicitly marginalize over the discrete variable y.

The final objective combines the supervised and unsupervised variational lower bound as well as the two auxiliary losses. Assuming N labeled and M unlabeled examples, we obtain the following objective:

$$\begin{aligned}\mathscr{F}_{\text{SS-DIVA}} = \sum_{n=1}^{N} \mathscr{F}_{\text{DIVA}}(d_n,\mathbf{x}_n,y_n) + \sum_{m=1}^{M} \mathscr{L}_u(d_m,\mathbf{x}_m) \\ + \alpha_d \mathbb{E}_{q_{\boldsymbol{\phi}_d}(\mathbf{z}_d|\mathbf{x}_m)}\left[\log q_{\boldsymbol{\omega}_d}(d_m|\mathbf{z}_d)\right].\end{aligned} \tag{5.4}$$

5.4 Related Work

The majority of proposed deep learning methods for domain generalization fall into one of two categories: 1) Learning a single domain invariant representation, e.g., using adversarial methods [Carlucci et al., 2019b, Ghifary et al., 2015, Li et al., 2018b,a, Motiian et al., 2017, Shankar et al., 2018, Wang et al., 2018]. While DIVA falls under this category, there is a key difference: we do not explicitly regularize $\mathbf{z}_y$ using d. Instead we learn complementary representations $\mathbf{z}_d$, $\mathbf{z}_x$ and $\mathbf{z}_y$ utilizing a generative architecture. 2) Ensembling models, each trained on an individual domain from the training set [Ding and Fu, 2018, Mancini et al., 2018]. The size of models in this category scales linearly with the number of training

domains. This leads to slow inference if the number of training domains is large. However, the size of DIVA is independent of the number of training domains. In addition, during inference time we only use the mean of the encoder $q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x})$ and the auxiliary classifier $q_{\boldsymbol{\omega}_y}(y|\mathbf{z}_y)$.

Concurrently to DIVA, Cai et al. [2019] developed a framework to learn latent Disentangled Semantic Representations (DSR) for domain adaptation. DSR assumes that the data generation process is exclusively controlled by the domain d and class y. As a result, DSR lacks a third latent space z_x. We designed DIVA assuming that not all variations in x can be explained by the domain d and the class y. Therefore we introduce z_x to capture these residual variations. Furthermore, while DSR uses gradient reversal layers, we directly parameterize the ground truth generative model.

An area that is closely related to domain generalization is that of the statistical parity in fairness. The goal of fair classification is to learn a meaningful representation that at the same time cannot be used to associate a data sample to a specific group [Zemel et al., 2013]. The significant difference to domain generalization is the intention behind that goal, e.g., to protect groups of individuals versus being robust to variations in the input. Consequently, DIVA is closely related to the fair VAE [Louizos et al., 2016]. In contrast to the fair VAE, which uses a hierarchical latent space, DIVA uses a partitioned latent space. Moreover, the fair VAE requires the domain label during inference, while DIVA alleviates this issue by learning the classifier without d. Similar to DIVA, there is an increasing number of methods showing the benefits of using latent subspaces in generative models [Siddharth et al., 2017, Klys et al., 2018, Jacobsen et al., 2018, Bouchacourt et al., 2018, Atanov et al., 2019, Antoran and Miguel, 2019].

We derived DIVA by following the VAE framework, where the generative process is the starting point. A Conditional version of the Variational Information Bottleneck (CVIB) was proposed by Moyer et al. [2018] that likewise leads to an objective consisting of a reconstruction loss. However, CVIB suffers from the same limitation as the fair VAE: the domain must be known during inference. Hence, we excluded it from our experiments.

5.5 Experiments

5.5.1 Rotated MNIST

The construction of the Rotated MNIST dataset follows Ghifary et al. [2015]. We sample 100 images from each of the ten classes from the original MNIST training dataset. This set of images is denoted $\mathcal{M}_{0^\circ}$. To create five additional domains, the images in $\mathcal{M}_{0^\circ}$ are rotated by 15, 30, 45, 60, and 75

degrees. In order to evaluate their domain generalization abilities, models are trained on five domains and tested on the remaining 6th domain, e.g., train on $\mathcal{M}_{0^\circ}$, $\mathcal{M}_{15^\circ}$, $\mathcal{M}_{30^\circ}$, $\mathcal{M}_{45^\circ}$ and $\mathcal{M}_{60^\circ}$, test on $\mathcal{M}_{75^\circ}$. The evaluation metric is the classification accuracy on the test domain. All experiments are repeated ten times. Detailed information about hyperparameters, architecture, and the training schedule can be found in the Appendix.

First, we visualize the three latent spaces $\mathbf{z}_d$, $\mathbf{z}_x$, and $\mathbf{z}_y$ to see if DIVA can successfully disentangle them. In addition, we want to verify whether DIVA utilizes $\mathbf{z}_x$ in a meaningful way since it is not directly connected to any auxiliary classifier. For now, we restrict the size of each latent space $\mathbf{z}_d$, $\mathbf{z}_x$, and $\mathbf{z}_y$ to two dimensions. Therefore, we can plot the latent subspaces directly without applying dimensionality reduction, see Figure 5.2, where we trained. DIVA on 5000 images from five domains: $\mathcal{M}_{0^\circ}$, $\mathcal{M}_{15^\circ}$, $\mathcal{M}_{30^\circ}$, $\mathcal{M}_{45^\circ}$ and $\mathcal{M}_{60^\circ}$.

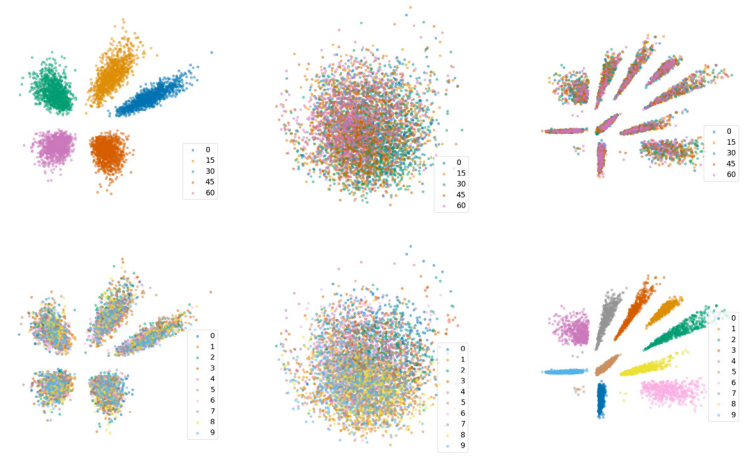


Figure 5.2. 2D embeddings of all three latent subspaces. In the top row, embeddings are colored according to their domain, in the bottom row they are colored according to their class. First column: $\mathbf{z}_d$ encoded by $q_{\boldsymbol{\phi}_d}(\mathbf{z}_d|\mathbf{x})$. The top plot shows five distinct clusters, where each cluster corresponds to a single domain. In the bottom plot, no clustering is visible. Second column: $\mathbf{z}_x$ encoded by $q_{\boldsymbol{\phi}_x}(\mathbf{z}_x|\mathbf{x})$. We observe a correlation between the rotation angle of each MNIST digit and $\mathbf{z}_x[0]$ in the top plot. Upon visual inspection of the original inputs $\mathbf{x}$, we find a correlation between the line thickness digit and $\mathbf{z}_x[0]$ as well as a correlation between the digit width and $\mathbf{z}_x[1]$ in the bottom plot. As a result, we observe the clustering of embeddings with class '1' at the lower left part of the plot. Third column: $\mathbf{z}_y$ encoded by $q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x})$. In the top plot, no clustering is visible. The bottom plot shows ten distinct clusters, where each cluster corresponds to a class.

Yet another way to gain insight into the disentanglement abilities of DIVA is conditional generation. We first train DIVA with $\beta = 10$ using $\mathcal{M}_{0^\circ}$, $\mathcal{M}_{15^\circ}$, $\mathcal{M}_{30^\circ}$, $\mathcal{M}_{45^\circ}$ and $\mathcal{M}_{60^\circ}$ as training domains. After training, we perform two experiments. In the first one, we are fixing the class and varying the

domain. In the second experiment, we are fixing the domain and varying the class.

Change of class The first row of Figure 5.3 (left) shows the input images x for DIVA. First, we generate embeddings $\mathbf{z}_d$, $\mathbf{z}_x$ and $\mathbf{z}_y$ for each $\mathbf{x}$ using $q_{\phi_d}(\mathbf{z}_d|\mathbf{x})$, $q_{\phi_x}(\mathbf{z}_x|\mathbf{x})$ and $q_{\phi_y}(\mathbf{z}_y|\mathbf{x})$. Second, we replace $\mathbf{z}_y$ with a sample $\mathbf{z}'_y$ from the conditional prior $p_{\theta_y}(\mathbf{z}_y|y)$. Last, we generate new images from $\mathbf{z}_d$, $\mathbf{z}_x$ and $\mathbf{z}'_y$ using the trained encoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_d, \mathbf{z}_x, \mathbf{z}_y)$. In Figure 5.3 (left) rows 2 to 11 correspond to the classes '0' to '9'. We observe that the rotation angle (encoded in $\mathbf{z}_d$) and the line thickness (encoded in $\mathbf{z}_x$) are well preserved, while the class of the image is changing as intended.

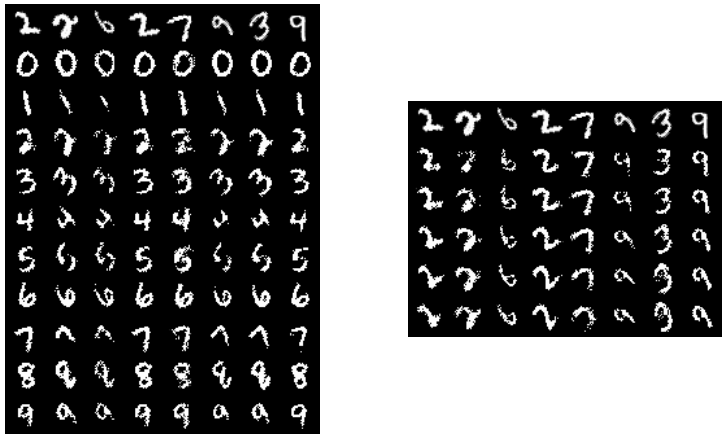


Figure 5.3. DIVA reconstructions. Left: First row is input, row 2 to 11 correspond to labels '0' to '9'. Right: First row is input, row 2 to 6 correspond to domains 0, 15, 30, 45, 60.

Change of domain We repeat the experiment from above but this time we keep $\mathbf{z}_x$ and $\mathbf{z}_y$ fixed while changing the domain. After generating embeddings $\mathbf{z}_d$, $\mathbf{z}_x$ and $\mathbf{z}_y$ for each $\mathbf{x}$ in the first row of Figure 5.3 (right), we replace $\mathbf{z}_d$ with a sample $\mathbf{z}'_d$ from the conditional prior $p_{\theta_d}(\mathbf{z}_d|d)$. Finally, we generate new images from $\mathbf{z}'_d$, $\mathbf{z}_x$ and $\mathbf{z}_y$ using the trained encoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_d, \mathbf{z}_x, \mathbf{z}_y)$. In Figure 5.3 (right) rows 2 to 6 correspond to the domains $\mathcal{M}_{0^\circ}$ to $\mathcal{M}_{60^\circ}$. Again, DIVA shows the desired behaviour: While the rotation angle is changing the class and style of the original image is maintained.

The qualitative results above conclude that DIVA is disentangling the information contained in $\mathbf{x}$ as intended, as $\mathbf{z}_d$ only contains information about d and $\mathbf{z}_y$ only information about y. In the case of the Rotated MNIST dataset, $\mathbf{z}_x$ captures any residual variation that is not explained by the domain d or the class y. We now turn to a quantitative evaluation of DIVA. We compare DIVA against the well known domain adversarial neural networks (DA) [Ganin et al., 2016] as well as three recently proposed

Table 5.1. Comparison with other state-of-the-art domain generalization methods. Methods in the first half of the table (until the vertical line) use only labeled data. The second half of the table shows results of DIVA when trained semi-supervised (+ X times the amount of unlabeled data). We report the average and standard error of the classification accuracy.

Test	DA	LG	HEX	ADV	DIVA	DIVA(+1)	DIVA(+3)	DIVA(+5)	DIVA(+9)
$\mathcal{M}_{0^\circ}$	86.7	89.7	90.1	89.9	**93.5 ± 0.3**	93.8 ± 0.4	93.9 ± 0.5	93.2 ± 0.5	93.0 ± 0.4
$\mathcal{M}_{15^\circ}$	98.0	97.8	98.9	98.6	**99.3 ± 0.1**	99.4 ± 0.1	99.5 ± 0.1	99.5 ± 0.1	99.6 ± 0.1
$\mathcal{M}_{30^\circ}$	97.8	98.0	98.9	98.8	**99.1 ± 0.1**	99.3 ± 0.1	99.3 ± 0.1	99.3 ± 0.1	99.3 ± 0.1
$\mathcal{M}_{45^\circ}$	97.4	97.1	98.8	98.7	**99.2 ± 0.1**	99.0 ± 0.2	99.2 ± 0.1	99.3 ± 0.1	99.3 ± 0.1
$\mathcal{M}_{60^\circ}$	96.9	96.6	98.3	98.6	**99.3 ± 0.1**	99.4 ± 0.1	99.4 ± 0.1	99.4 ± 0.1	99.2 ± 0.2
$\mathcal{M}_{75^\circ}$	89.1	92.1	90.0	90.4	**93.0 ± 0.4**	93.8 ± 0.4	93.8 ± 0.2	93.5 ± 0.4	93.2 ± 0.3
Avg	94.3	95.3	95.8	95.2	**97.2 ± 1.3**	97.5 ± 1.1	97.5 ± 1.2	97.4 ± 1.3	97.3 ± 1.3

methods: LG [Shankar et al., 2018], HEX [Wang et al., 2018] and ADV [Wang et al., 2018]. For the first half of Table 5.1 (until the vertical line), we only use labeled data. The first column indicates the rotation angle of the test domain. We report test accuracy on y for all methods. For DIVA, we report the mean and standard error for ten repetitions. DIVA achieves the highest accuracy across all test domains and the highest average test accuracy among all proposed methods. The second half of Table 5.1 showcases the ability of DIVA to use unlabeled data. For this experiment, we add the same amount (+1) of unlabeled data as well as three (+3), five (+5), and nine (+9) times the amount of unlabeled data to our training set. We first add the unlabeled data to $\mathcal{M}_{0^\circ}$ and create the data for the other domains. In Table 5.1 we can see a performance increase when unlabeled data is added to the training set. When the number of unlabeled data is much larger than the number of labeled data, the balancing of loss terms becomes increasingly more challenging, which can lead to declining performance of DIVA, as seen in the last two columns of Table 5.1.

In the experiment described above, each training domain consists of labeled and unlabeled examples. We investigate a more challenging scenario: We add a domain to our training set composed of only unlabeled examples. Regarding our introductory example of medical imaging, here, we would add unlabeled data from a new patient or hospital to the training set. In the following, we are looking at two different experimental setups. In both cases, $\mathcal{M}_{75^\circ}$ is the test domain: For the first experiment we choose the domains $\mathcal{M}_{0^\circ}$, $\mathcal{M}_{15^\circ}$, $\mathcal{M}_{45^\circ}$ and $\mathcal{M}_{60^\circ}$ to be part of the labeled training set. In addition, unlabeled data from $\mathcal{M}_{30^\circ}$ is used. We find that even in the case where the additional domain is dissimilar to the test domain, DIVA can slightly improve, see Table 5.2. For the second experiment we choose the domains $\mathcal{M}_{0^\circ}$, $\mathcal{M}_{15^\circ}$, $\mathcal{M}_{30^\circ}$ and $\mathcal{M}_{45^\circ}$ to be part of the labeled training set. Also, unlabeled data from $\mathcal{M}_{60^\circ}$ is used. When comparing with the results in Table 5.1, we notice a drop in accuracy of about 20% for DIVA trained with only labeled data. However, when trained with unlabeled data from

$\mathcal{M}_{60°}$ we see an improvement of about 7%, see Table 5.2. The comparison shows that DIVA can successfully learn from samples of a domain without any labels.

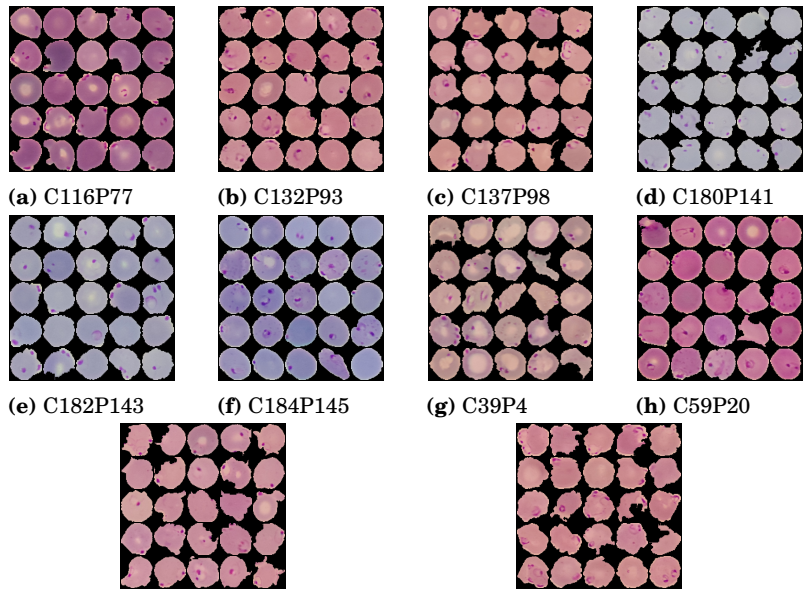
Table 5.2. Comparison of DIVA trained supervised to DIVA trained semi-supervised with additional unlabeled data from $\mathcal{M}_{30°}$ and $\mathcal{M}_{60°}$. We report the average and standard error of the classification accuracy on $\mathcal{M}_{75°}$.

Unsupervised domain	DIVA supervised	DIVA semi-supervised
$\mathcal{M}_{30°}$	93.1 ± 0.5	93.3 ± 0.4
$\mathcal{M}_{60°}$	73.8 ± 0.8	**80.6** ± **1.1**

5.5.2 Malaria Cell Images

The majority of medical imaging datasets consist of images from a multitude of patients. In a domain generalization setting, each patient is viewed as an individual domain. While we focus on *patients as domains* in this paper, this type of reasoning can be extended to, e.g., *hospitals as domains*. We, among others [Rajaraman et al., 2018, Lafarge et al., 2017], argue that machine learning algorithms trained with medical imaging datasets should be evaluated on a subset of holdout patients. This presents a more realistic scenario since the algorithm is tested on images from a previously unseen domain. In the following, we use a Malaria Cell Images dataset [Rajaraman et al., 2018] as an example of a dataset consisting of samples from multiple patients. The images in this dataset were collected and photographed at Chittagong Medical College Hospital, Bangladesh. It consists of 27558 single red blood cell images taken from 150 infected and 50 healthy patients. A human expert manually annotated the images. A cell has the label $y = 1$ if it shows the parasite and the label $y = 0$ if not. To facilitate the counting of parasitized and uninfected cells, the cells were stained using Giemsa stain, which turns the parasites inside the cell pink. Besides, the staining process leads to a variety of colors of the cell itself. While the color of the cell is relatively constant for a single patient, it can vary significantly between patients, see Figure 5.4. A human observer can easily ignore this variability in the appearance of the cells. However, machine learning models can fail to generalize across patients. In our experiments, we will use the patient ID as the domain label d. We argue that for this specific dataset, the patient ID is a good proxy of appearance variability. In addition, there is no extra cost for obtaining the patient ID for each cell. Subsequently, we use a subset of the Malaria Cell Images dataset consisting of the ten patients with the highest number of cells. The amount of cells per patient varies between 400 and 700, and there are 5922 cell images in total. The choice of this subset is motivated by a

similar number of cells as well as the similar marginal label distributions per patient, the latter being a necessary condition for successful domain generalization Zhao et al. [2019]. Furthermore, we rescale all images to 64 × 64 pixels. To artificially expand the size of the training dataset, we use data augmentation in the form of vertical flips, horizontal flips, and random rotations.



(a) C116P77 **(b)** C132P93 **(c)** C137P98 **(d)** C180P141

(e) C182P143 **(f)** C184P145 **(g)** C39P4 **(h)** C59P20

(i) C68P29 **(j)** C99P60

Figure 5.4. Example cells from 10 patients of the Malaria Cell Images dataset.

We investigate the three latent subspaces $\mathbf{z}_d$, $\mathbf{z}_x$, and $\mathbf{z}_y$ to see if DIVA can successfully disentangle them. In addition, we want to see if DIVA utilizes $\mathbf{z}_x$ in a meaningful way since it is not directly connected to any auxiliary classifier. Figure 5.5 shows the reconstructions of $\mathbf{x}$ using all three latent subspaces as well as reconstructions of $\mathbf{x}$ using only a single latent subspace at a time. First, we find that DIVA can reconstruct the original cell images using all three subspaces (Figure 5.5, second row). Second, we find that the three latent subspaces are indeed disentangled: $\mathbf{z}_d$ is containing the color of the cell (Figure 5.5, third row), $\mathbf{z}_x$ the shape of the cell (Figure 5.5, fourth row) and $\mathbf{z}_y$ the location of the parasite (Figure 5.5, fifth row). The holes in the reconstructions using only $\mathbf{z}_x$ indicate that there is no probability mass in $\mathbf{z}_d$ and $\mathbf{z}_y$ at 0, similar to Figure 5.2. From the reconstructions in Figure 5.5, we conclude that DIVA can learn disentangled representations that match the ground truth factors of interest, here, the appearance of the cell and the presence of the parasite.

Models are trained on nine domains (patient IDs) and tested on the

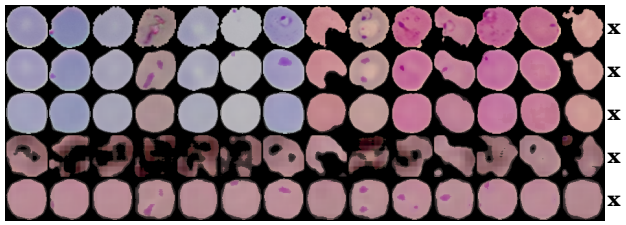


Figure 5.5. Reconstructions of $\mathbf{x}$ using all three latent subspaces as well as reconstructions of $\mathbf{x}$ using only a single latent subspace at a time.

remaining 10th domain to further evaluate domain generalization abilities. We choose ROC AUC on the holdout test domain as the evaluation metric since the two classes are highly imbalanced. All experiments are repeated five times. We compare DIVA with a ResNet-like [He et al., 2016] baseline and DA. All three models have the same architecture during inference, seven ResNet blocks followed by two linear layers. Detailed information about hyperparameters, architecture, and the training schedule can be found in the Appendix. We compare DIVA against a ResNet (Baseline), a domain adversarial neural networks (DA), and HEX [Wang et al., 2018]. We find that the results are not equally distributed across all test domains. In five cases, DIVA significantly improves upon the baseline model, DA, and HEX. However, averaged over all domains, none of the four methods performs significantly better than the others, see Table 5.3. We find that while HEX can achieve excellent generalization performance, it is not stable for different seeds. The large standard error reflects this for each of the test domains.

Table 5.3. Comparison with other state-of-the-art domain generalization methods on the Malaria Cell image dataset. We report the average and standard error of ROC AUC.

Model	C116P77	C132P93	C137P98	C180P141	C182P143	C184P145
Baseline	90.6 ± 0.7	97.8 ± 0.5	98.9 ± 0.2	98.5 ± 0.2	96.7 ± 0.4	98.1 ± 0.2
DA	90.6 ± 1.7	**98.3 ± 0.4**	99.0 ± 0.1	98.8 ± 0.1	96.9 ± 0.4	97.1 ± 0.8
HEX	76.33 ± 10.38	80.58 ± 7.86	77.80 ± 4.38	73.22 ± 8.94	89.38 ± 2.39	63.81 ± 7.04
DIVA	**93.3 ± 0.4**	**98.4 ± 0.3**	99.0 ± 0.1	**99.0 ± 0.1**	96.5 ± 0.3	**98.5 ± 0.3**

Model	C39P4	C59P20	C68P29	C99P60	Average
Baseline	97.1 ± 0.4	82.8 ± 2.8	95.3 ± 0.6	96.2 ± 0.1	95.2 ± 1.6
DA	97.4 ± 0.3	83.2 ± 3.3	**96.3 ± 0.1**	96.1 ± 0.3	95.4 ± 1.6
HEX	72.84 ± 7.58	81.52 ± 7.80	67.88 ± 7.15	84.99 ± 3.48	76.8 ± 2.5
DIVA	**97.8 ± 0.2**	82.1 ± 3.0	**96.3 ± 0.2**	**96.6 ± 0.3**	95.8 ± 1.6

As described in Section 5.3.1, we are interested in learning from domains with no class labels since such an approach can drastically lower the amount of labeled data needed to learn a domain invariant representation, i.e., a model that generalizes well across patients. For the semi-supervised

experiments, we choose domain C116P77 to be the test domain since its cells show a unique dark pink stain. Furthermore, unlabeled data from domain C59P20 is used since it is visually the closest to domain C116P77, see Figure 5.4. The evaluation metric on the hold-out test domain is ROC AUC again. In Table 5.4 we compare the baseline model, DA, and DIVA trained with labeled data from domain C59P20, unlabeled data from domain C59P20, and no data from domain C59P20. We argue that the improvement of DIVA over DA arises from the way the additional unlabeled data is utilized. In the case of DA, the unlabeled data (d, $\mathbf{x}$) is only used to train the domain classifier and the feature extractor in an adversarial manner. In Section 5.3.1 we show that due to DIVA's generative nature $q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x})$, $p_{\boldsymbol{\theta}_y}(\mathbf{z}_y|y)$ can be updated using $q_{\boldsymbol{\omega}_y}(y|\mathbf{z}_y)$ to marginalize over y for an unlabeled sample $\mathbf{x}$. In addition, the unlabeled data (d, $\mathbf{x}$) is used to update $q_{\boldsymbol{\phi}_d}(\mathbf{z}_d|\mathbf{x})$, $p_{\boldsymbol{\theta}_d}(\mathbf{z}_d|d)$, $q_{\boldsymbol{\omega}_d}(d|\mathbf{z}_d)$, $q_{\boldsymbol{\phi}_x}(\mathbf{z}_x|\mathbf{x})$ and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_d, \mathbf{z}_x, \mathbf{z}_y)$ in the same way as in the supervised case.

Table 5.4. Results of the semi-supervised experiments for domain C116P77. Comparison of baseline method, DA and DIVA trained with labeled data from domain C59P20, unlabeled data from domain C59P20, and no data from domain C59P20. We report the average and standard error of ROC AUC.

Training data	Baseline	DA	DIVA
Labeled data from C59P20	90.6 ± 0.7	90.6 ± 1.7	**93.3 ± 0.4**
Unlabeled data from C59P20	-	72.05 ± 2.2	**79.4 ± 2.8**
No data from C59P20	70.0 ± 2.6	69.2 ± 1.9	71.9 ± 2.7

5.6 Conclusion

We have proposed DIVA as a generative model with three latent subspaces. We evaluated DIVA on Rotated MNIST and a Malaria Cell Images dataset. In both cases, DIVA can learn disentangled representations that match the ground truth factors of interest, represented by the class y and the domain d. By learning representations, $\mathbf{z}_y$ that are invariant to the domain d DIVA improves upon other methods on both datasets. Furthermore, we show that we can boost DIVA's performance by incorporating unlabeled samples, even from entirely new domains for which no labeled examples are available. This property is highly desirable in fields like medical imaging, where the labeling process is very time-consuming and costly. It appears that there is a key difference between interpolation and extrapolation, a distinction currently not made by the domain generalization community. If we assume that the domains lie in intervals like [0°,15°, 30°] or ['red', 'orange', 'yellow'] then the performance for the domains in the center of the interval, e.g., 15° and 'orange', seems to be better than for the domains on the ends of the interval. We argue that DIVA can use unlabeled data from a domain close

to the test domain to improve its extrapolation performance.

5.7 Appendix

5.7.1 Predicting the label using only one of the latent subspaces

We test how predictive $\mathbf{z}_d$, $\mathbf{z}_x$, and $\mathbf{z}_y$ are for the class y on the Malaria Cell Images dataset. First, we use the trained DIVA models from 5.5.2 to create embeddings $\mathbf{z}_d$, $\mathbf{z}_x$ and $\mathbf{z}_y$ for every $\mathbf{x}$ in the training domain and the holdout test domain. Second, we train a 2-layer MLP on the embeddings $\mathbf{z}_d$, $\mathbf{z}_x$, and $\mathbf{z}_y$ from the training domains. We train the MLP for 100 epochs using ADAM [Kingma and Ba, 2015]. After training, we test the MLP embeddings $\mathbf{z}_d$, $\mathbf{z}_x$, and $\mathbf{z}_y$ from the test domain. In Table 5.5 we see that $\mathbf{z}_y$ captures all relevant information to predict y, while the MLPs trained using $\mathbf{z}_d$ and $\mathbf{z}_x$ perform worse than a classifier that would always pick the majority class.

Table 5.5. Prediction of y using a 2 layer MLP trained using $\mathbf{z}_d$, $\mathbf{z}_x$ and $\mathbf{z}_y$. We report the mean and standard error of the classification accuracy on the hold out test domain.

test domain	$\mathbf{z}_d$	$\mathbf{z}_x$	$\mathbf{z}_y$	majority class
0	84.6 ± 1.0	85.0 ± 0.2	**87.9 ± 0.9**	0.86
1	89.5 ± 0.4	88.2 ± 0.5	**96.8 ± 0.1**	0.9
2	68.2 ± 3.5	80.0 ± 1.6	**96.9 ± 0.5**	0.81
3	87.0 ± 0.3	75.2 ± 2.9	**95.5 ± 0.2**	0.88
4	89.1 ± 0.3	82.7 ± 2.4	**92.5 ± 0.4**	0.90
5	88.3 ± 0.2	87.7 ± 0.2	**90.6 ± 0.5**	0.88
6	82.6 ± 3.7	56.3 ± 5.1	**91.1 ± 0.1**	0.90
7	88.3 ± 0.1	88.3 ± 0.1	**90.8 ± 0.8**	0.88
8	89.5 ± 0.3	85.3 ± 1.7	**93.5 ± 0.4**	0.90
9	89.1 ± 0.2	86.9 ± 1.5	**94.0 ± 0.3**	0.89

5.7.2 Partitioned latent space

We compare DIVA to a VAE with a single latent space, a standard Gaussian prior, and two auxiliary tasks. The resulting graphical model is shown in Figure 5.6. The results in Table 5.6 clearly show the benefits of having a partitioned latent space $\mathbf{z}$.

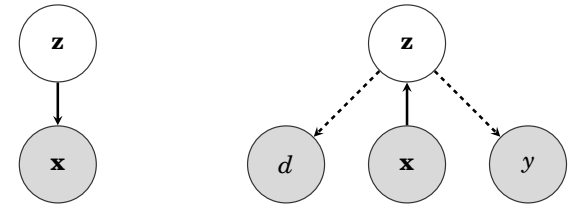


Figure 5.6. DAG of an VAE with auxiliary classifiers. Left: Generative model. According to the graphical model we obtain $p(\mathbf{x},\mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})$. Right: Inference model. We propose $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ as the variational posterior. Dashed arrows represent the two auxiliary classifiers $q_{\boldsymbol{\omega}_d}(d|\mathbf{z})$ and $q_{\boldsymbol{\omega}_y}(y|\mathbf{z})$.

The objective is given by,

$$\begin{aligned}\mathscr{F}_{\text{VAE}}(d,\mathbf{x},y) := \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - \beta KL\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right) \\ + \alpha_d \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log q_{\boldsymbol{\omega}_d}(d|\mathbf{z})\right] + \alpha_y \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log q_{\boldsymbol{\omega}_y}(y|\mathbf{z})\right]. \qquad (5.5)\end{aligned}$$

Table 5.6. Comparison of DIVA with a VAE with a single latent space, a standard Gaussian prior and two auxillary tasks on Rotated MNIST. We report the average and standard error of the classification accuracy.

Test	VAE	DIVA
$\mathscr{M}_{0°}$	88.4 ± 0.5	**93.5 ± 0.3**
$\mathscr{M}_{15°}$	98.3 ± 0.1	**99.3 ± 0.1**
$\mathscr{M}_{30°}$	97.4 ± 0.2	**99.1 ± 0.1**
$\mathscr{M}_{45°}$	97.4 ± 0.4	**99.2 ± 0.1**
$\mathscr{M}_{60°}$	97.9 ± 0.2	**99.3 ± 0.1**
$\mathscr{M}_{75°}$	84.0 ± 0.3	**93.0 ± 0.4**
Avg	93.9 ± 0.1	**97.2 ± 1.3**

5.7.3 DIVA without the domain latent subspace or the residual latent subspace

We compare DIVA as proposed in Section 5.3 to two ablated versions of DIVA:

1. DIVA without z_d: The domain label d is not used during training. Therefore, there exist no latent space z_d, no encoder $q_{\boldsymbol{\phi}_d}(z_d|x)$, no prior $p_{\boldsymbol{\theta}_d}(z_d|d)$ and no classifier $q_{\boldsymbol{\omega}_d}(d|z_d)$. The decoder becomes $p_{\boldsymbol{\theta}}(x|z_x,z_y)$.

2. DIVA without z_x: There exist no latent space z_x, no encoder $q_{\boldsymbol{\phi}_x}(z_x|x)$ and no prior $p(z_x)$. The decoder becomes $p_{\boldsymbol{\theta}}(x|z_d,z_y)$.

In Table 5.7, we compare DIVA and the two ablated versions on the Rotated MNIST dataset. Surprisingly, for Rotated MNIST, we could not find

a significant difference in performance between DIVA and DIVA without z_d, as seen in the third column. However, not having z_d drastically reduces the interpretability of our model since, without z_d, we cannot find the variations in x that are explained by the domain d. E.g., in Figure 5.5, we show that we can generate samples conditioned on the domain label that give us a clear idea of the meaning of d. We find that the patient ID is highly correlated with the color of the stain. While the cell shape is not correlated with d or y and therefore is captured by z_x. Without z_d, we cannot gain such (especially from a medical perspective) important insights. In the fourth column, we see that for $\mathcal{M}_{0^\circ}$ and $\mathcal{M}_{75^\circ}$ DIVA with z_x performs significantly better than without. We argue that if z_x does not exist, z_d and z_y will capture the residual variations in x that are not explained by d or y. We believe this makes it harder to predict y using z_y and d using z_d.

Table 5.7. Results of ablation study.

Test	DIVA	DIVA without z_d	DIVA without z_x
$\mathcal{M}_{0^\circ}$	93.5 ± 0.3	93.4 ± 0.5	92.7 ± 0.5
$\mathcal{M}_{15^\circ}$	99.3 ± 0.1	99.3 ± 0.1	99.4 ± 0.1
$\mathcal{M}_{30^\circ}$	99.1 ± 0.1	98.9 ± 0.1	99.2 ± 0.1
$\mathcal{M}_{45^\circ}$	99.2 ± 0.1	99.1 ± 0.1	99.1 ± 0.1
$\mathcal{M}_{60^\circ}$	99.3 ± 0.1	99.1 ± 0.1	99.4 ± 0.1
$\mathcal{M}_{75^\circ}$	93.0 ± 0.4	92.8 ± 0.4	92.4 ± 0.4
Avg	97.2 ± 1.3	97.1 ± 1.3	97.1 ± 1.5

5.8 Experiment details

5.8.1 Rotated MNIST

Training procedure All DIVA models are trained for 500 epochs. The training is terminated if the training loss for y has not improved for 100 epochs. As proposed in Burgess et al. [2018], we linearly increase β from 0.0 to 1.0 during the first 100 epochs of training. We set α_d = 2000. As seen in Maaløe et al. [2019], we adjust α_y according to the ratio of labeled (N) and unlabeled data (M),

$$\alpha_y = \gamma \frac{M+N}{N}, \tag{5.6}$$

where we set $\gamma = 3500$. Last, $\mathbf{z}_d$, $\mathbf{z}_x$ and $\mathbf{z}_y$ each have 64 latent dimensions. All hyperparameters were determined by training DIVA on $\mathcal{M}_{0^\circ}$, $\mathcal{M}_{15^\circ}$, $\mathcal{M}_{30^\circ}$, $\mathcal{M}_{45^\circ}$ and testing on $\mathcal{M}_{60^\circ}$. We searched over the following parameters:

α_d, $\alpha_d \in \{1500, 2000, 2500, 3000, 3500, 4000\}$; $\dim(\mathbf{z}_d) = \dim(\mathbf{z}_x) = \dim(\mathbf{z}_y)$ and $\dim(\mathbf{z}_x) \in \{16, 32, 64\}$; $\beta_{max} \in \{1, 5, 10\}$.

All models were trained using ADAM [Kingma and Ba, 2015] (with default settings), a pixel-wise cross-entropy loss, and a batch size of 100.

Architectures To enable a fair experiment, the encoder $q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x})$ and auxiliary classifier $q_{\boldsymbol{\omega}_y}(y|\mathbf{z}_y)$ form a DNN with the same number of layers and weights as described in Wang et al. [2018].

Table 5.8. Architecture for $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_d, \mathbf{z}_x, \mathbf{z}_y)$. The parameter for Linear is output features. The parameters for ConvTranspose2d are output channels and kernel size. The parameter for Upsample is the upsampling factor. The parameters for Conv2d are output channels and kernel size.

block	details
1	Linear(1024), BatchNorm1d, ReLU
2	Upsample(2)
3	ConvTranspose2d(128, 5), BatchNorm2d, ReLU
4	Upsample(2)
5	ConvTranspose2d(256, 5), BatchNorm2d, ReLU
6	Conv2d(256, 1)

Table 5.9. Architecture for $p_{\boldsymbol{\theta}_d}(\mathbf{z}_d|d)$ and $p_{\boldsymbol{\theta}_y}(\mathbf{z}_y|y)$. Each network has two heads one for the mean and one for the scale. The parameter for Linear is output features.

block	details
1	Linear(64), BatchNorm1d, ReLU
2.1	Linear(64)
2.2	Linear(64), Softplus

Table 5.10. Architecture for $q_{\boldsymbol{\phi}_d}(\mathbf{z}_d|\mathbf{x})$, $q_{\boldsymbol{\phi}_x}(\mathbf{z}_x|\mathbf{x})$ and $q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x})$. Each network has two heads one for the mean one and for the scale. The parameters for Conv2d are output channels and kernel size. The parameters for MaxPool2d are kernel size and stride. The parameter for Linear is output features.

block	details
1	Conv2d(32, 5), BatchNorm2d, ReLU
2	MaxPool2d(2, 2)
3	Conv2d(64, 5), BatchNorm2d, ReLU
4	MaxPool2d(2, 2)
5.1	Linear(64)
5.2	Linear(64), Softplus

Table 5.11. Architecture for $q_{\boldsymbol{\omega}_d}(d|\mathbf{z}_d)$ and $q_{\boldsymbol{\omega}_y}(y|\mathbf{z}_y)$. The parameter for Linear is output features.

block	details
1	ReLU, Linear(5 for $q_{\boldsymbol{\omega}_d}(d|\mathbf{z}_d)$/10 for $q_{\boldsymbol{\omega}_y}(y|\mathbf{z}_y)$), Softmax

5.8.2 Malaria Cell Images

Training procedure: DIVA All DIVA models are trained for 500 epochs. The training is terminated if the validation accuracy for y has not improved for 100 epochs. As proposed in Burgess et al. [2018], we linearly increase β from 0.0 to 1.0 during the first 100 epochs of training. We set α_d = 100000 and α_y = 75000. Last, $\mathbf{z}_d$, $\mathbf{z}_x$ and $\mathbf{z}_y$ each have 64 latent dimensions. We searched over the following parameters: α_d, $\alpha_d \in \{25000, 50000, 75000, 100000\}$; $\dim(\mathbf{z}_d) = \dim(\mathbf{z}_x) = \dim(\mathbf{z}_y)$, $\dim(\mathbf{z}_x) \in \{32, 64\}$; $\beta_{max} \in \{1, 5, 10\}$. All hyperparameters were determined using a validation set that consists of 20 % of the training set. All models were trained using ADAM [Kingma and Ba, 2015] (with default settings), a mixture of discretized logistics [Salimans et al., 2017] loss and a batch size of 100. In case of the semi-supervised experiment in Section 5.5.2 we adapt α_d and α_y according to Equation 5.6.

Training procedure: Baseline and DA In the case of the supervised experiments in Section 5.5.2, all models are trained for 500 epochs. The training is terminated if the validation accuracy for y has not improved for 100 epochs. In the case of the semi-supervised experiments in Section 5.3.1, the amount of epochs is adjusted to match the number of parameter updates of DIVA. For DA, we follow the same training procedure as described in Ganin et al. [2016]. In the supervised case, a labeled batch randomly sampled from the training distributions is used to update the class classifier, domain classifier, and feature extractor in an adversarial fashion. Second, a second batch randomly sampled from the training distributions is used to update only the domain classifier and the feature extractor in an adversarial fashion. In the semi-supervised case, samples from the unsupervised domains from the second batch and samples from the supervised domains. We use the same domain adaptation parameter λ schedule as described in Ganin et al. [2016]. Determined by hyperparameter search, DA performs better when $\lambda \cdot \epsilon$ is used. Here, $\epsilon = 0.001$. We searched over the following values of $\epsilon \in \{0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001\}$. In case of the semi-supervised experiment in Section 5.5.2 $\epsilon = 0.01$ was determined by hyperparameter search.

Architecture In the following, we will describe the architecture of DIVA in detail. Note that the architecture for the baseline model is the same as $q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x})$ (we only use the mean of $\mathbf{z}_y$) followed by $q_{\boldsymbol{\omega}_y}(y|\mathbf{z}_y)$ where $\mathbf{z}_y$ has 1024 dimensions. DA is using $q_{\boldsymbol{\phi}_y}(\mathbf{z}_y|\mathbf{x})$ without the linear layer as a feature extractor. The class classifier and the domain classifier consist of two linear layers. The feature extractor for all models consists of seven ResNet blocks [He et al., 2016]. During training, batch norm Ioffe and Szegedy [2016] is used for all layers.

Table 5.12. Architecture for $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_d, \mathbf{z}_x, \mathbf{z}_y)$. The parameter for Linear is output features. The parameters for ResidualConvTranspose2d are output channels and kernel size. The parameters for Conv2d are output channels and kernel size.

block	details
1	Linear(1024), BatchNorm1d, LeakyReLU
2	ResidualConvTranspose2d(64, 3), LeakyReLU
3	ResidualConvTranspose2d(64, 3), LeakyReLU
4	ResidualConvTranspose2d(64, 3), LeakyReLU
5	ResidualConvTranspose2d(32, 3), LeakyReLU
6	ResidualConvTranspose2d(32, 3), LeakyReLU
7	ResidualConvTranspose2d(32, 3), LeakyReLU
8	ResidualConvTranspose2d(32, 3), LeakyReLU
9	ResidualConvTranspose2d(32, 3), LeakyReLU
10	Conv2d(100, 3)
11	Conv2d(100, 1)

Table 5.13. Architecture for $p_{\boldsymbol{\theta}_d}(\mathbf{z}_d|d)$ and $p_{\boldsymbol{\theta}_y}(\mathbf{z}_y|y)$. Each network has two heads one for the mean and one for the scale. The parameter for Linear is output features.

block	details
1	Linear(64), BatchNorm1d, LeakyReLU
2.1	Linear(64)
2.2	Linear(64), Softplus

Table 5.14. Architecture for $q_{\phi_d}(\mathbf{z}_d|\mathbf{x})$, $q_{\phi_x}(\mathbf{z}_x|\mathbf{x})$ and $q_{\phi_y}(\mathbf{z}_y|\mathbf{x})$. Each network has two heads one for the mean one and for the scale. The parameters for Conv2d are output channels and kernel size. The parameters for ResidualConv2d are output channels and kernel size. The parameter for Linear is output features.

block	details
1	Conv2d(32, 3), BatchNorm2d, LeakyReLU
2	ResidualConv2d(32), LeakyReLU
3	ResidualConv2d(32), LeakyReLU
4	ResidualConv2d(64, 3), LeakyReLU
5	ResidualConv2d(64, 3), LeakyReLU
6	ResidualConv2d(64, 3), LeakyReLU
7	ResidualConv2d(64, 3), LeakyReLU
8	ResidualConv2d(64, 3), LeakyReLU
9.1	Linear(64)
9.2	Linear(64), Softplus

Table 5.15. Architecture for $q_{\omega_d}(d|\mathbf{z}_d)$ and $q_{\omega_y}(y|\mathbf{z}_y)$. The parameter for Linear is output features.

block	details
1	LeakyReLU, Linear(9 for $q_{\omega_d}(d|\mathbf{z}_d)$/2 for $q_{\omega_y}(y|\mathbf{z}_y)$), Softmax

6. Conclusion

In each of the previous four chapters, we have encountered a different experimental setup, where each experimental setup required a different approach to train invariant machine learning models. In Chapter 2, we use an attention-pooling layer to build a permutation invariant deep learning architecture. In Chapter 3, we use data augmentation to obtain invariance to symmetries described by (semi)group transformations like rotation, reflection, and color. In Chapter 4, we use normalizing flows to learn invariant mechanisms from interventional and observational data. Last, in Chapter 5, we use an augmented VAE to learn domain invariant features by disentanglement.

We argue that there is a hierarchy of the difficulty of the experimental setups. The difficulty arises from the amount of expert knowledge we use in each of the four cases. We consider knowing the exact symmetry group of the data and task a large amount of expert knowledge. This type of knowledge is usually derived from decades of research in respective fields like chemistry and physics. On the other hand, using additional information like *metadata* [1] to group the training data into domains, we consider the least amount of background knowledge.

Furthermore, we argue that a trade-off exists between the "quality" of the invariance and the amount of expert knowledge used. Where more expert knowledge leads to strict invariance and less expert knowledge leads to approximate invariance. We summarize the above arguments in a ladder of invariances.

1. The symmetry of the data and task is known and can be expressed as a group and therefore enforced using a specifically designed architecture. The resulting machine learning model is strictly invariant. This approach is often used in chemistry and physics applications, where the underlying symmetry of, e.g., molecules or a system of particles is known. One of the most recent examples of this approach is AlphaFold 2, a deep learning

[1] Metadata refers to additional information about a sample, e.g., in a healthcare application we would refer to age, sex, and ethnicity as metadata.

model for protein structure prediction [Jumper et al., 2021]. An SE(3)-Transformer is used to ensure equivariance with respect to rotation and translation.

2. The symmetry of the data and task is known but cannot be expressed as a group. In this case, instead of a specifically designed architecture, we can often use data augmentation. However, the resulting machine learning model will only be approximately invariant. This approach is commonly used for images, where invariance with respect to semigroup transformations like color perturbations and scaling is favorable. We want to highlight that recent progress in self-supervised learning [Chen et al., 2020b] heavily relies on data augmentation, preventing the model from overfitting.

3. The symmetry of the data and task is unknown, but we have access to experimental samples with known interventions from, e.g., RCTs. This approach allows the machine learning model to learn independent causal mechanisms directly. However, experimental data is often not fully unbiased [Mansournia et al., 2017]. Therefore strict invariance of the machine learning model is not guaranteed. While this approach is commonly used to estimate the efficiency of treatments such as medical drugs, there is a growing interest in the reinforcement learning community [Zhang et al., 2020].

4. The symmetry of the data and task is unknown. In addition, we have no access to experimental samples with known interventions. Instead, we have data from multiple domains, where for example, metadata can be used to cluster the data. This approach assumes that a machine learning model that generalizes across training domains will generalize to previous unseen domains. The success of such an approach depends on the variety of the machine learning model's training data and inductive bias. Therefore, it comes with no guarantees that the model will be invariant after training. This approach is commonly used with medical data from different hospitals or self-driving car data collected in multiple countries.

Our so-called ladder of invariance shows that there is currently no one-size-fits-all approach for training invariant machine learning models. Instead, in most cases, extensive expert knowledge is required to obtain a machine learning model with the correct type of invariance, where expert knowledge can often be represented in the form of a symmetry group or a causal graph, see Section 1.2.1 and 1.2.2.

Unifying the two view points In Section 1, we formally introduced two notions of invariance. One is derived from the concepts of symmetry

groups, the other from invariant causal mechanisms. We explicitly connect data augmentations and interventions in Chapter 3, where we defined data augmentations as (semi)group transformations, thus connecting symmetry groups to causal inference.

In addition, we have seen a more implicit connection of symmetry groups and causal inference in Chapter 5. To train DIVA, we require data from different domains. We assume a different set of interventions was performed for each domain, i.e., each domain is equal to a different interventional distribution. To obtain domain invariant features, we rely on the disentanglement of the latent space, where disentanglement is defined using the concept of group symmetry, see Section 1.3.1. We implicitly find a connection of the encoder that maps to a domain invariant subspace and the true invariant causal mechanism that generalizes across domains.

The above connection of disentangled representation and invariant causal mechanisms motivates the following extension of the intervention-augmentation equivariance condition from Chapter 3 to include learned invariant representations.

We consider the same setup as in Chapter 3, Figure 3.2. We assume that there are high-level features $\mathbf{h}_y$ caused by the target variable Y and high-level features $\mathbf{h}_d$ caused by nuisance variables, e.g., the domain variable D. Furthermore, we call the function $f_{\mathbf{X}} : \mathscr{H}_d \times \mathscr{H}_y \to \mathscr{X}$ the true causal mechanism.

In addition, we define a set of transformations G on $\mathscr{X}$ that form a (semi)group, where the group actions g can be interpreted as data augmentations, i.e., $\mathbf{x}_{\text{aug}} = g \cdot \mathbf{x}$.

We now extend our setup with a learned representation $\mathbf{Z} \in \mathscr{Z}$, an encoder $f_{\boldsymbol{\theta}} : \mathscr{X} \to \mathscr{Z}$ with trainable parameters $\boldsymbol{\theta}$, and a classifier $h_{\boldsymbol{\phi}} : \mathscr{Z} \to \mathscr{Y}$ with trainable parameters $\boldsymbol{\phi}$. Furthermore, we assume that there exists a corresponding action on $\mathscr{Z}$, $g \cdot \mathbf{z} : G \times \mathscr{Z} \to \mathscr{Z}$, where we denote the $\mathbf{z}_g = g \cdot \mathbf{z}$.

We call the function $f_{\boldsymbol{\theta}}$ an equivariant map if it commutes with a group action $g \in G$, i.e.,

$$f_{\boldsymbol{\theta}}(g \cdot \mathbf{x}) = g \cdot f_{\boldsymbol{\theta}}(\mathbf{x}). \tag{6.1}$$

We can rewrite the LHS and the RHS using the definitions of $\mathbf{x}_{\text{aug}}$ and $\mathbf{z}$ as follows

$$f_{\boldsymbol{\theta}}(\mathbf{x}_{\text{aug}}) = g \cdot \mathbf{z}. \tag{6.2}$$

Last, using $\mathbf{z}_g = g \cdot \mathbf{z}$ we obtain

$$f_{\boldsymbol{\theta}}(\mathbf{x}_{\text{aug}}) = \mathbf{z}_g. \tag{6.3}$$

Above, we have demonstrated that first applying data augmentation followed by $f_{\boldsymbol{\theta}}$ results in the same representation as computing the embedding $\mathbf{z}$ first and then applying a group action $g \in G$.

Throughout the present thesis, we have seen multiple examples of such equivariant maps $f_{\boldsymbol{\theta}}$. The most obvious one being group-equivariant neural networks in Section 1.2.1.

In addition, we have seen two examples of the special case where the $f_{\boldsymbol{\theta}}$ is an invariant map, i.e., $f_{\boldsymbol{\theta}}(g \cdot \mathbf{x}) = f_{\boldsymbol{\theta}}(\mathbf{x})$, which is equivalent to $\mathbf{z}_g = \mathbf{z}$. The first example is the attention-pooling layer introduced in Chapter 2, that maps a set to a representation $\mathbf{Z}$ invariant to the set's permutations. The second example is the encoder of an VAE that maps to a disentangled representation as seen in Chapter 5. For simplicity, we now assume that the disentangled latent space consists of two subspaces $(\mathbf{z}, \mathbf{z}')$, where the subspace $\mathbf{z}$ is invariant with respect to group actions $g \in G$. We can now define $f_{\boldsymbol{\theta}}$ as the part of the encoder that maps from $\mathbf{x}$ to $\mathbf{z}$, excluding $\mathbf{z}'$.

Last we define a classifier $h_{\boldsymbol{\phi}}$ that maps $\mathbf{z}$ to the target variable y. In Figure 6.1, we show a commutative diagram that relates all of the elements above.

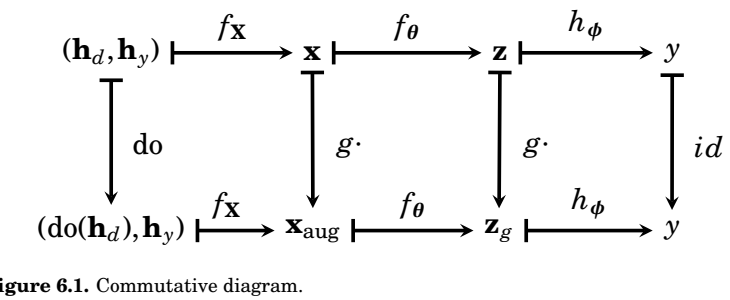


Figure 6.1. Commutative diagram.

As seen in Chapter 3, we find that augmenting the data commutes with intervention before data generation. In addition, we see that data augmentation commutes with the group action g on $\mathbf{z}$, a result that is at the basis of research concerning group-equivariant neural networks. Last, we are able to connect equivariant representations $\mathbf{z}$ to interventions prior to data generation. Intuitively, $f_{\boldsymbol{\theta}}$ inverts parts of the causal mechanism $f_{\mathbf{X}}$ and is, therefore, able to recover the true high-level features $\mathbf{h}_y$ that were initially caused by Y. In summary, the commutative diagram in Figure 6.1 connects all concepts we have encountered throughout the present thesis: interventions, causal mechanisms, data augmentation, group-invariant, and group-equivariant representations.

Acknowledgements

I want to thank my supervisor Max Welling for providing the opportunity to pursue a PhD and for his guidance and advice.

I want to express my sincere gratitude to my co-promoter, Jakub Tomczak. When I arrived at the AMLab, I knew very little about machine learning. Nonetheless, you took your time to teach me how to be a successful scientist. I owe you a lot.

After Jakub left the AMLab, Patrick Forré became my new mentor. Patrick showed me how to be rigorous without overly complicating things. A skill that I will cherish for the rest of my career. Thank you.

I want to thank Chris Louizos and Joris Mooij for collaborating with me. I learned a lot from the two of you. Without you, this thesis would not exist.

Many thanks to my colleagues and friends Karen, Bas, Rianne, Marco, Jorn, Emiel, Andy, Daniel, Thomas, Wouter, Matthias, and Sarah for their topical and non-topical advice.

Thank you to Felice Arends, Dennis Koelma, Bas Terwijn for their help with administration and IT.

Thank you to Ben Glocker, Nick Pawlowski, and Daniel Castro for the numerous discussions about causality and medical imaging.

Thank you to every member of DLMedIA and Bart Bakker of Philips research. Our meetings helped me to ground my work in real medical applications.

Last, thank you to everybody at Microsoft health intelligence for the excellent internship experience.

References

Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 16: Annual Conference on Neural Information Processing Systems 2003, NeurIPS 2003*, 2003.

Joshua D. Angrist, Guido W. Imbens, and Donald B. Rubin. Identification of Causal Effects Using Instrumental Variables. *Journal of the American Statistical Association*, 91(434):444–455, 1996.

Javier Antoran and Antonio Miguel. Disentangling and Learning Robust Representations with Natural Clustering. *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant Risk Minimization. *arXiv:1907.02893 [cs, stat]*, 2020.

Andrei Atanov, Alexandra Volokhova, Arsenii Ashukha, Ivan Sosnovik, and Dmitry Vetrov. Semi-Conditional Normalizing Flows for Semi-Supervised Learning. *arXiv:1905.00505 [cs, stat]*, 2019.

Susan Athey, Raj Chetty, and Guido Imbens. Combining Experimental and Observational Data to Estimate Treatment Effects on Long Term Outcomes. *arXiv:2006.09676 [cs, stat]*, 2020.

Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *Journal of Machine Learning Research 20*, 2019.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*, 2014.

Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. MetaReg: Towards Domain Generalization using Meta-Regularization. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, 2018.

Alexander Balke and Judea Pearl. Bounds on Treatment Effects From Studies With Imperfect Compliance. *Journal of the American Statistical Association*, 92 (439):1171–1176, 1997.

Elias Bareinboim and Judea Pearl. Causal inference and the data-fusion problem. *Proceedings of the National Academy of Sciences*, 113(27):7345–7352, 2016.

J. S. Bell. On the Einstein Podolsky Rosen paradox. *Physics, Physique, Fizika*, 1 (3):6, 1964.

Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 20:1–6, 2019.

Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from Several Related Classification Tasks to a New Unlabeled Sample. In *Advances in Neural Information Processing Systems 24: Annual Conference on Neural Information Processing Systems 2011, NeurIPS 2011*, 2011.

Stephan Bongers, Patrick Forré, Jonas Peters, Bernhard Schölkopf, and Joris M. Mooij. Foundations of Structural Causal Models with Cycles and Latent Variables. *arXiv:1611.06221*, 2020.

Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-Level Variational Autoencoder: Learning Disentangled Representations from Grouped Observations. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.

Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-VAE. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, 2018.

Ruichu Cai, Zijian Li, Pengfei Wei, Jie Qiao, Kun Zhang, and Zhifeng Hao. Learning Disentangled Semantic Representation for Domain Adaptation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Macao, China, 2019.

Fabio M. Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain Generalization by Solving Jigsaw Puzzles. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019a.

Fabio Maria Carlucci, Paolo Russo, Tatiana Tommasi, and Barbara Caputo. Hallucinating Agnostic Images to Generalize Across Domains. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, Seoul, Korea (South), 2019b.

Daniel C. Castro, Ian Walker, and Ben Glocker. Causality matters in medical imaging. *Nature Communications*, 11(1):3673, 2020.

Shuxiao Chen, Edgar Dobriban, and Jane H Lee. A Group-Theoretic Framework for Data Augmentation. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020a.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, 2020b.

Yixin Chen, Jinbo Bi, and James Ze Wang. MILES: Multiple-instance learning via embedded instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):1931–1947, 2006.

Veronika Cheplygina, Lauge Sørensen, David MJ Tax, Marleen de Bruijne, and Marco Loog. Label stability in multiple instance learning. In *MICCAI*, pages 539–546, 2015a.

Veronika Cheplygina, David MJ Tax, and Marco Loog. Multiple instance learning with bag dissimilarities. *Pattern Recognition*, 48(1):264–275, 2015b.

Francesco Ciompi, Oscar Geessink, Babak Ehteshami Bejnordi, Gabriel Silva de Souza, Alexi Baidoshvili, Geert Litjens, Bram van Ginneken, Iris Nagtegaal, and Jeroen van der Laak. The importance of stain normalization in colorectal tissue classification with convolutional networks. In *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pages 160–163, Melbourne, Australia, 2017.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016*, 2016.

Bénédicte Colnet, Imke Mayer, Guanhua Chen, Awa Dieng, Ruohong Li, Gaël Varoquaux, Jean-Philippe Vert, Julie Josse, and Shu Yang. Causal inference methods for combining randomized trials and observational studies: a review. *arXiv:2011.08047*, 2020.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian Neural Networks. *arXiv:2003.04630 [physics, stat]*, 2020.

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning Augmentation Strategies From Data. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 113–123, Long Beach, CA, USA, 2019.

Bin Dai and David Wipf. Diagnosing and Enhancing VAE Models. *arXiv:1903.05789 [cs, stat]*, 2019.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, 2017.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71, 1997.

Zhengming Ding and Yun Fu. Deep Domain Generalization With Structured Low-Rank Constraint. *IEEE Transactions on Image Processing*, 27(1):304–313, 2018.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

Hadi Mohaghegh Dolatabadi, Sarah M. Erfani, and Christopher Leckie. Invertible generative modeling using linear rational splines. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020*, 2020.

Gary Doran and Soumya Ray. A theoretical and empirical analysis of support vector machine methods for multiple-instance classification. *Machine Learning*, 97(1-2):79–102, 2014.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, 2019.

Ji Feng and Zhi-Hua Zhou. Deep miml network. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 1884–1890. AAAI Press, 2017.

Patrick Forré. Transitional Conditional Independence. *arxiv:2104.11547*, 2021.

Richard D. Fuhr and Michael Kallay. Monotone linear rational spline interpolation. *Computer Aided Geometric Design*, 9(4):313–319, 1992.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17:1–35, 2016.

Thomas Gärtner, Peter A Flach, Adam Kowalczyk, and Alexander J Smola. Multi-instance kernels. In *Proceedings of the 19th International Conference on Machine Learning, ICML 2002*, 2002.

Elisa Drelie Gelasca, Jiyun Byun, Boguslaw Obara, and BS Manjunath. Evaluation and benchmark for biological image segmentation. In *IEEE International Conference on Image Processing*, pages 1816–1819, 2008.

Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain Generalization for Object Recognition with Multi-task Autoencoders. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010.

Raphael Gontijo-Lopes, Sylvia J. Smullin, Ekin D. Cubuk, and Ethan Dyer. Affinity and Diversity: Quantifying Mechanisms of Data Augmentation. *arXiv:2002.08973 [cs, stat]*, 2020.

Sven Gowal, Chongli Qin, Po-Sen Huang, Taylan Cemgil, Krishnamurthy Dvijotham, Timothy Mann, and Pushmeet Kohli. Achieving Robustness in the Wild via Adversarial Mixing With Disentangled Representations. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1208–1217, 2020.

Florian Gunsilius. A path-sampling method to partially identify causal effects in instrumental variable models. *arxiv:1910.09502*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, USA, 2016.

Christina Heinze-Deml and Nicolai Meinshausen. Conditional variance penalties and domain shift robustness. *Machine Learning*, 110(2):303–348, 2021.

I. Higgins, L. Matthey, A. Pal, Christopher P. Burgess, Xavier Glorot, M. Botvinick, S. Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a Definition of Disentangled Representations. *arXiv:1812.02230 [cs, stat]*, 2018.

Keisuke Hirano and Guido W. Imbens. The Propensity Score with Continuous Treatments. In *Wiley Series in Probability and Statistics*, pages 73–84. Wiley, 2005.

Le Hou, Dimitris Samaras, Tahsin M Kurc, Yi Gao, James E Davis, and Joel H Saltz. Patch-based convolutional neural network for whole slide tissue image classification. In *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Maximilian Ilse, Jakub M Tomczak, Christos Louizos, and Max Welling. DIVA: Domain Invariant Variational Autoencoders. *Medical Imaging with Deep Learning*, page 27, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2016*, 2016.

Jörn-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive Invariance Causes Adversarial Vulnerability. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

Fredrik D. Johansson, David Sontag, and Rajesh Ranganath. Support and Invertibility in Domain-Invariant Representations. *AISTATS 2019*, 2019.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, July 2021.

Nathan Kallus, Aahlad Manas Puli, and Uri Shalit. Removing hidden confounding by experimental grounding. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, 2018.

Melih Kandemir and Fred A Hamprecht. Computer-aided diagnosis from weak supervision: a benchmarking study. *Computerized Medical Imaging and Graphics*, 42:44–50, 2015.

Melih Kandemir, Chong Zhang, and Fred A Hamprecht. Empowering multiple instance histopathology cancer diagnosis by cell graphs. In *MICCAI*, pages 228–235, 2014.

Melih Kandemir, Manuel Haußmann, Ferran Diego, Kumar T Rajamani, Jeroen van der Laak, and Fred A Hamprecht. Variational Weakly Supervised Gaussian Processes. In *BMVC*, 2016.

James D Keeler, David E Rumelhart, and Wee Kheng Leow. Integrated segmentation and recognition of hand-printed numerals. In *Advances in Neural Information Processing Systems 4: Annual Conference on Neural Information Processing Systems 1991, NeurIPS 1991*, 1991.

Ilyes Khemakhem, Ricardo Pio Monti, Robert Leech, and Aapo Hyvärinen. Causal Autoregressive Flows. In *arXiv:2011.02268*, 2020.

Aditya Khosla, Tinghui Zhou, Tomasz Malisiewicz, Alexei A. Efros, and Antonio Torralba. Undoing the Damage of Dataset Bias. In *ECCV*, Berlin, Heidelberg, 2012.

Niki Kilbertus, Matt J Kusner, and Ricardo Silva. A class of algorithms for general instrumental variable models. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, pages 20108–20119, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2th International Conference on Learning Representations, ICLR 2013*, 2013.

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. Semi-Supervised Learning with Deep Generative Models. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, NeurIPS 2014*, 2014.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

Jack Klys, Jake Snell, and Richard Zemel. Learning Latent Subspaces in Variational Autoencoders. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, 2018.

I. Kobyzev, S. Prince, and M. Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

Oren Z Kraus, Jimmy Lei Ba, and Brendan J Frey. Classifying and segmenting microscopy images with deep multiple instance learning. *Bioinformatics*, 32 (12):i52–i59, 2016.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25: Annual Conference on Neural Information Processing Systems 2012, NeurIPS 2012*, 2012.

David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Remi Le Priol, and Aaron Courville. Out-of-Distribution Generalization via Risk Extrapolation (REx). *arXiv:2003.00688 [cs, stat]*, 2020.

Maxime W. Lafarge, Josien P. W. Pluim, Koen A. J. Eppenhof, Pim Moeskops, and Mitko Veta. Domain-adversarial neural networks to address the appearance variability of histopathology images. *arXiv:1707.06183 [cs]*, 2017.

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

Sanghack Lee, Juan D. Correa, and Elias Bareinboim. General identifiability with arbitrary surrogate experiments. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI-19)*, volume 115, pages 389–398. PMLR, 2020.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, Broader and Artier Domain Generalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to Generalize: Meta-Learning for Domain Generalization. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018a.

Sharon Y. Li. Automating Data Augmentation: Practice, Theory and New Direction, 2020. Library Catalog: ai.stanford.edu.

Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep Domain Generalization via Conditional Invariant Adversarial Networks. In *ECCV*, 2018b.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60 – 88, 2017.

Guoqing Liu, Jianxin Wu, and Zhi-Hua Zhou. Key instance detection in multi-instance learning. In *Journal of Machine Learning Research*, volume 25, pages 253–268, 2012.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, 2019.

Christos Louizos, Kevin Swersky, Yujia Li, M. Welling, and R. Zemel. The variational fair autoencoder. *CoRR*, abs/1511.00830, 2016.

Christos Louizos, Uri Shalit, Joris M. Mooij, David A. Sontag, Richard S. Zemel, and Max Welling. Causal effect inference with deep latent-variable models. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, NeurIPS 2017*, 2017.

Clare Lyle, Mark van der Wilk, Marta Kwiatkowska, Yarin Gal, and Benjamin Bloem-Reddy. On the Benefits of Invariance in Neural Networks. *arXiv:2005.00178 [cs, stat]*, 2020.

Lars Maaløe, Marco Fraccaro, Valentin Liévin, and Ole Winther. BIVA: A Very Deep Hierarchy of Latent Variables for Generative Modeling. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, 2019.

David Madigan, Paul E. Stang, Jesse A. Berlin, Martijn Schuemie, J. Marc Overhage, Marc A. Suchard, Bill Dumouchel, Abraham G. Hartzema, and Patrick B. Ryan. A systematic statistical approach to evaluating evidence from observational studies. *Annual Review of Statistics and Its Application*, 1(1): 11–39, 2014.

Sara Magliacane, Thijs van Ommen, Tom Claassen, Stephan Bongers, Philip Versteeg, and Joris M Mooij. Domain Adaptation by Using Causal Inference to Predict Invariant Conditional Distributions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10846–10856. Curran Associates, Inc., 2018.

Massimiliano Mancini, S. R. Bulò, B. Caputo, and E. Ricci. Best sources forward: Domain generalization through source-specific nets. *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 1353–1357, 2018.

Mohammad Ali Mansournia, Julian P. T. Higgins, Jonathan A. C. Sterne, and Miguel A. Hernán. Biases in Randomized Trials: A Conversation Between Trialists and Epidemiologists. *Epidemiology*, 28(1):54–59, January 2017.

Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems 11: Annual Conference on Neural Information Processing Systems 1998, NeurIPS 1998*, 1998.

Wang Miao, Zhi Geng, and Eric J Tchetgen Tchetgen. Identifying causal effects with proxy variables of an unmeasured confounder. *Biometrika*, 105(4):987–993, 2018.

Joris M. Mooij, Jonas Peters, Dominik Janzing, Jakob Zscheischler, and Bernhard Schölkopf. Distinguishing Cause from Effect Using Observational Data: Methods and Benchmarks. *Journal of Machine Learning Research*, 17:1–102, 2016.

Joris M. Mooij, Sara Magliacane, and Tom Claassen. Joint causal inference from multiple contexts. *Journal of Machine Learning Research*, 21(99):1–108, 2020.

Saeid Motiian, Marco Piccirilli, Donald A. Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

Daniel Moyer, Shuyang Gao, Rob Brekelmans, Greg Ver Steeg, and Aram Galstyan. Invariant Representations without Adversarial Training. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, 2018.

Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain Generalization via Invariant Feature Representation. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, 2013.

Emmy Noether and M. A. Tavel. Invariant Variation Problems. *Transport Theory and Statistical Physics*, 1(3):186–207, January 1971.

Avital Oliver, Augustus Odena, Colin Raffel, Ekin Dogus Cubuk, and Ian J. Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, 2018.

Maxime Oquab, Léon Bottou, Ivan Laptev, Josef Sivic, et al. Weakly supervised object recognition with convolutional neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, NeurIPS 2014*, 2014.

George Papamakarios, Iain Murray, and Theo Pavlakou. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, 2017.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. In *Journal of Machine Learning Research*, volume 22, pages 1–64, 2021.

Nikolaos Pappas and Andrei Popescu-Belis. Explaining the stars: Weighted multiple-instance learning for aspect-based sentiment analysis. In *EMNLP*, pages 455–466, 2014.

Nikolaos Pappas and Andrei Popescu-Belis. Explicit Document Modeling through Weighted Multiple-Instance Learning. *Journal of Artificial Intelligence Research*, 58:591–626, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, 2019.

Nick Pawlowski, Daniel Coelho de Castro, and Ben Glocker. Deep structural causal models for tractable counterfactual inference. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

Judea Pearl. On the testability of causal models with latent and instrumental variables. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, 1995.

Judea Pearl. *Causality*. Cambridge University Press, 2 edition, 2009.

Luis Perez and Jason Wang. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *arXiv:1712.04621*, 2017.

Jonas Peters, Peter Bühlmann, and Nicolai 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.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, Cambridge, MA, USA, 2017.

Pedro O Pinheiro and Ronan Collobert. From image-level to pixel-level labeling with convolutional networks. In *2015 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Gwenole Quellec, Guy Cazuguel, Beatrice Cochener, and Mathieu Lamard. Multiple-instance learning for medical image and video analysis. *IEEE Reviews in Biomedical Engineering*, 2017.

Colin Raffel and Daniel P. W. Ellis. Feed-Forward Networks with Attention Can Solve Some Long-Term Memory Problems. *arXiv:1512.08756 [cs]*, 2016.

Sivaramakrishnan Rajaraman, Sameer K. Antani, Mahdieh Poostchi, Kamolrat Silamut, Md. A. Hossain, Richard J. Maude, Stefan Jaeger, and George R. Thoma. Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. *PeerJ*, 6:e4568, 2018.

Jan Ramon and Luc De Raedt. Multi instance neural networks. In *ICML Workshop on Attribute-value and Relational Learning*, 2000.

Vikas C Raykar, Balaji Krishnapuram, Jinbo Bi, Murat Dundar, and R Bharat Rao. Bayesian multiple instance learning: automatic feature selection and inductive transfer. In *Proceedings of the 25th International Conference on Machine Learning, ICML 2008*, 2008.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning, ICML 2014*, 2014.

Lucia Ricci-Vitiani, Dario G Lombardi, Emanuela Pilozzi, Mauro Biffoni, Matilde Todaro, Cesare Peschle, and Ruggero De Maria. Identification and expansion of human colon-cancer-initiating cells. *Nature*, 445(7123):111, 2007.

Mateo Rojas-Carulla, Bernhard Scholkopf, Richard Turner, and Jonas Peters. Invariant Models for Causal Transfer Learning. *Journal of Machine Learning Research*, 19:1–34, 2018.

Evan Rosenman, Art B. Owen, Michael Baiocchi, and Hailey Banack. Propensity Score Methods for Merging Observational and Experimental Datasets. *arXiv:1804.07863*, 2018.

Evan Rosenman, Guillaume Basse, Art Owen, and Michael Baiocchi. Combining Observational and Experimental Datasets Using Shrinkage Estimators. *arXiv:2002.06708*, 2020.

Arnout C Ruifrok and Dennis A Johnston. Quantification of histochemical staining by color deconvolution. *Analytical and Quantitative Cytology and Histology*, 23 (4):291–299, 2001.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. *arXiv:1701.05517 [cs, stat]*, 2017. arXiv: 1701.05517.

Stephen Scott, Jun Zhang, and Joshua Brown. On generalized multiple-instance learning. *International Journal of Computational Intelligence and Applications*, 5(01):21–35, 2005.

Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing Across Domains via Cross-Gradient Training. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

N. Siddharth, Brooks Paige, Jan-Willem van de Meent, Alban Desmaison, Noah D. Goodman, Pushmeet Kohli, Frank Wood, and Philip H. S. Torr. Learning Disentangled Representations with Semi-Supervised Deep Generative Models. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, NeurIPS 2017*, 2017.

Ricardo Silva. Observational-interventional priors for dose-response learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, NeurIPS 2016*, 2016.

Korsuk Sirinukunwattana, Shan E Ahmed Raza, Yee-Wah Tsang, David RJ Snead, Ian A Cree, and Nasir M Rajpoot. Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images. *IEEE Transactions on Medical Imaging*, 35(5):1196–1206, 2016.

Adarsh Subbaswamy and Suchi Saria. From development to deployment: dataset shift, causality, and shift-stable models in health AI. *Biostatistics*, pages 0–41, 2019.

Adarsh Subbaswamy, Peter Schulam, and Suchi Saria. Preventing Failures Due to Dataset Shift: Learning Predictive Models That Transport. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019.

E. G. Tabak and Cristina V. Turner. A Family of Nonparametric Density Estimation Algorithms. *Communications on Pure and Applied Mathematics*, 66(2): 145–164, 2013.

David Tellez, Geert Litjens, Péter Bándi, Wouter Bulten, John-Melle Bokhorst, Francesco Ciompi, and Jeroen van der Laak. Quantifying the effects of data augmentation and stain color normalization in convolutional neural networks for computational pathology. *Medical Image Analysis*, 2019.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep Domain Confusion: Maximizing for Domain Invariance. *arXiv:1412.3474*, 2014.

V. Vapnik. Principles of Risk Minimization for Learning Theory. In *Advances in Neural Information Processing Systems 5: Annual Conference on Neural Information Processing Systems 1992, NeurIPS 1992*, 1992.

Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual Attention Network for Image Classification. In *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Haohan Wang, Zexue He, Zachary C. Lipton, and Eric P. Xing. Learning Robust Representations by Projecting Superficial Statistics Out. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

Xinggang Wang, Yongluan Yan, Peng Tang, Xiang Bai, and Wenyu Liu. Revisiting multiple instance neural networks. *Pattern Recognition*, 74:15–24, 2016.

Yixin Wang and David M. Blei. The blessings of multiple causes. *Journal of the American Statistical Association*, 114(528):1574–1596, 2019.

Xiu-Shen Wei, Jianxin Wu, and Zhi-Hua Zhou. Scalable algorithms for multi-instance learning. *IEEE Transactions on Neural Networks and Learning Systems*, 28(4):975–987, 2017.

Elie Wolfe, Robert W. Spekkens, and Tobias Fritz. The inflation technique for causal inference with latent variables. *Journal of Causal Inference*, 7(2), 2019.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, 2015.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep Sets. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, NeurIPS 2017*, 2017.

Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 325–333, Atlanta, Georgia, USA, 2013. PMLR.

Cha Zhang, John C Platt, and Paul A Viola. Multiple instance boosting for object detection. In *Advances in Neural Information Processing Systems 19: Annual Conference on Neural Information Processing Systems 2006, NeurIPS 2006*, 2006.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond Empirical Risk Minimization. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

Junzhe Zhang, Daniel Kumor, and Elias Bareinboim. Causal imitation learning with unobserved confounders. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

Qi Zhang and Sally A Goldman. Em-dd: An improved multiple-instance learning technique. In *Advances in Neural Information Processing Systems 15: Annual Conference on Neural Information Processing Systems 2002, NeurIPS 2002*, 2002.

Han Zhao, Remi Tachet des Combes, Kun Zhang, and Geoffrey J. Gordon. On Learning Invariant Representation for Domain Adaptation. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, 2019.

Zhi-Hua Zhou, Yu-Yin Sun, and Yu-Feng Li. Multi-instance learning by treating instances as non-iid samples. In *Proceedings of the 26th International Conference on Machine Learning, ICML 2009*, 2009.

Wentao Zhu, Qi Lou, Yeeleng Scott Vang, and Xiaohui Xie. Deep multi-instance networks with sparse label assignment for whole mammogram classification. In *MICCAI*, pages 603–611, 2017.